MASCODE : un système multi-agent adaptatif pour concevoir des produits complexes. Application à la conception préliminaire avion.

THÈSE

présentée et soutenue publiquement le 31 mars 2008

pour l'obtention du

Doctorat de l'Université de Toulouse

(délivré par l'Université Paul Sabatier, mention informatique et intelligence artificielle)

par

Jean-Baptiste WELCOMME

Composition du jury

Rapporteurs :	Olivier BOISSIER, *Professeur, École des Mines de St Étienne*
	René MANDIAU, *Professeur, Université de Valenciennes*
Examinateurs :	Pierre De LOOR, *Maître de conférence HdR, ENI de Brest*
	Michel DUREIGNE, *Ingénieur, EADS Innovation Works Suresnes*
	Marie-Pierre GLEIZES, *Professeur, Directrice de thèse, Université Paul Sabatier*
	Jean-Baptiste HIRIART-URRUTY, *Professeur, Université Paul Sabatier*
	Romaric REDON, *Ingénieur, EADS Innovation Works Toulouse*
Invités :	Sylvie DELPRAT, *Ingénieur, EADS Innovation Works Toulouse*
	Thierry DRUOT, *Ingénieur, Airbus France Toulouse*
	Pierre GLIZE, *Ingénieur CNRS HdR, IRIT Toulouse*
	Philippe MATTEI, *Ingénieur, SOGETI France Toulouse*

Institut de recherche en informatique de Toulouse - UMR 5505

Mis en page avec la classe thloria.

Table des matières

Table des figures

Introduction générale

1 Le contexte et l'analyse de l'existant

Lors de la conception d'un avion de nombreux facteurs doivent être pris en compte, tels que :

- le degré d'innovation technologique du produit ;
- la complexité de son architecture ;
- le nombre d'alternatives de conception disponible ;
- la disparité des connaissances et des compétences disciplinaires à mettre en oeuvre ;
- la réponse aux besoins des compagnies aériennes ;
- la présence de fournisseurs et de sous-traitants qualifiés, etc.

Le rôle de la conception avion est donc aujourd'hui de prendre en compte un maximum de facteurs, de les analyser et de proposer des configurations d'avions qui permettent de répondre au mieux à un ensemble de besoins, en identifiant plusieurs alternatives de conception, les points qui seront délicats à traiter lors des phases de conception plus détaillées, etc.

En parallèle à cette complexification, les délais de conception doivent être de plus en plus courts et maîtrisés afin de répondre le plus rapidement possible aux besoins changeant des compagnies aériennes, qui évoluent sur un secteur d'activité très concurrentiel et soumis à des instabilités.

Dans ce contexte, les rôles et les enjeux de la conception préliminaire avion sont donc renforcés et accrus. Elle est aujourd'hui traitée comme étant un problème d'optimisation multi-disciplinaire et multi-objectif, pour lequel on cherche à trouver un ensemble de valeurs de paramètres caractérisant l'avion. En général, cette démarche utilise des méthodes classiques d'optimisation, qui traitent le problème d'une manière globale en proposant des solutions satisfaisant l'ensemble des contraintes. Mais ces méthodes restent limitées, car elles ne prennent pas en compte l'ensemble de la structure du problème. Par exemple, elles ignorent les interactions entre les disciplines et les composants impliqués dans la conception avion. De ce fait, elles se montrent limités dès que le nombre d'objectif devient significatif ou que la structure du problème se complexifie.

Pour pallier ce problème, nous avons identifié l'ensemble des exigences fonctionnelles qui permettent d'améliorer la conception préliminaire avion. Elles consistent principalement à apporter de nouvelles solutions technologiques, qui permettent de mieux considérer les propriétés du problème en prenant en compte simultanément les aspects suivants :

- Cognitifs (définition des connaissances propres à chaque modèle) : la conception avion nécessite la mise en commun de modèles disciplinaires. Chaque modèle représente une fonction induite de la physique générale avion, pour laquelle on a des connaissances qui permettent de la définir, de la choisir, de l'ajuster, etc. Il est donc nécessaire de prendre en compte ces connaissances si l'on veut avoir une bonne définition et compréhension multi-disciplinaire et multi-physique du problème global.
- Logiques d'interaction (intégration des exigences par coopération) : la mise en relation des modèles nécessite la prise en compte des contraintes et des exigences de chacun d'entre eux. Pour illustrer les relations et interdépendances qui lient les modèles, il faut donc pouvoir proposer des logiques de négociation aboutissant à des solutions satisfaisantes pour chacun d'entre eux.
- Distribution physique (hétérogénéité et autonomie des modèles) : chaque modèle disciplinaire est conçu indépendamment et susceptible de changer en fonction des évolutions et découpages fonctionnelles de la conception. Les exécutions, exigences et préférences de chaque modèle doivent donc être gérées localement et intégrer dynamiquement par des échanges d'information.

Tenir compte de ces aspects cognitifs, logiques et physiques en même temps et de cette manière parait judicieux, puisque une logique par interaction permet de prendre en compte les contraintes physiques due à la distribution en utilisant les connaissances propres à chaque modèle.
Grâce à leurs qualités d'ouverture, d'adaptation dynamique, les systèmes multi-agents sont une solution technologique en adéquation avec les propriétés générales du problème. La solution que nous proposons est donc basée sur un système multi-agent adaptatif dans lequel les agents représentent les disciplines, les paramètres de conception et les performances de l'avion.

2 Les contributions

Les recherches menées dans cette thèse ont conduit à des contributions portant sur deux domaines :

- la conception préliminaire avion ;
- la résolution de problèmes multi-disciplinaires par un système multi-agent adaptatif.

Après avoir justifié la nécessité d'utiliser de nouvelles technologies pour traiter les besoins industriels, nous proposons un algorithme de régulation de contraintes, qui tient compte de la structure du problème et des connaissances associées aux modèles de simulation utilisés en conception avant-projet avion. Des agents, en utilisant des connaissances disciplinaires et par un comportement coopératif, trouvent collectivement les valeurs des paramètres

de conception qui satisfont les contraintes et les performances. Notre approche utilise un système dans lequel les agents ont des connaissances individuelles et un raisonnement purement local, ce qui permet d'obtenir un système conceptuellement distribué et ayant des propriétés d'adaptation et d'auto-organisation.
En utilisant ces propriétés du système, nous montrons qu'il est facile de le compléter et de l'outiller pour répondre aux besoins spécifiques de la conception préliminaire avion. Dans ce cadre, nous proposons un processus d'optimisation et des moyens pour le concepteur d'interagir dynamiquement avec le système.

2.1 La démarche de la recherche

L'environnement

Ces travaux ont commencé en avril 2005 et ont été menés au sein d'EADS Innovation Works (IW) qui a pour but de réaliser des études de recherche industrielle pour l'ensemble des entreprises du groupe (Airbus, Eurocopter, Astrium, MBDA, EADS DCS, etc.), et en partenariat avec l'IRIT (Institut de Recherche en Informatique de Toulouse) et Airbus France.

Le phasage de la recherche

Lors de notre recherche, nous avons commencé par identifier les problématiques générales de la conception avion ainsi que l'ensemble des limites actuelles. Cette étude a montré qu'elles sont principalement dues à un manque de considération de l'ensemble des dimensions du problème (aspects multi-niveaux et multi-disciplinaires) dans les outils actuels. Suite à l'identification de ces manques, nous avons proposé une démarche de recherche utilisant les systèmes multi-agents. Pour répondre à cette démarche, nous avons choisi de commencer dans un premier temps par proposer un système d'auto-régulation de contraintes. Puis pour valider notre approche et notre démarche, nous nous sommes ensuite concentrés sur l'ajout de fonctionnalités orientées métiers autour de ce nouveau système. Cette seconde phase, nous a permis de comparer notre approche aux solutions existantes ou en cours de recherche. Ainsi, nous montrons nos apports et les bénéfices de notre approche.

2.2 Organisation du travail de thèse

Notre recherche a été organisée de la manière suivante :

1. définir les besoins industriels, justifier les besoins technologiques et proposer une problématique de recherche ;

2. définir le cadre théorique de notre recherche ;
3. proposer une nouvelle approche de résolution multi-disciplinaire, basée sur un système multi-agent adaptatif ;
4. développer et expérimenter cette nouvelle approche pour répondre à des besoins précis de la conception avion ;
5. évaluer les apports et limites de nos propositions.

3 Le plan

Ce document est composé de 3 parties, contenant plusieurs chapitres. Le plan retenu est illustré par la figure 1. Afin de faciliter la lecture du document, les parties 1 et 2 sont précédées et terminées par de courtes introductions/conclusions.

1. La première partie du document est constituée de 3 chapitres, définissant le contexte et la problématique de recherche.
2. Les chapitres de la deuxième partie présentent les principaux axes de la recherche et détaillent les propositions.
3. La dernière partie est consacrée à l'application des propositions, à la comparaison des résultats sur des problèmes de conception préliminaire avion, ainsi qu'à la description des perspectives à ce travail.

Part 1 - Le contexte industriel et le positionnement scientifique
Chapitre 1 : La conception avion et ses besoins fonctionnels
Chapitre 2 : L'analyse des solutions existantes
Chapitre 3 : Le contexte théorique et scientifique
Part 2 - Les contributions
Chapitre 4 : Un SMA adaptatif pour l'auto-régulation sous-contraintes
Chapitre 5 : L'optimisation des solutions
Chapitre 6 : La co-construction des solutions avec l'utilisateur
Part 3 -Application des propositions à la conception préliminaire avion
Chapitre 7 : L'application MASCODE
Chapitre 8 : Les résultats et les performances de MASCODE
Chapitre 9 : Les perspectives

FIG. 1 – Plan de la thèse

Première partie

Le contexte industriel et le positionnement scientifique

Introduction

Les travaux présentés dans cette thèse portent sur l'amélioration des outils d'aide à la conception préliminaire avion.
Afin de s'approprier la problématique, le premier chapitre décrit le contexte de réalisation de l'étude, définissant les spécificités du produit avion et du secteur industriel. Ce premier chapitre se termine par une illustration des étapes principales de la conception avion et par l'identification de besoins fonctionnels qu'une ou plusieurs applications informatiques pourraient apporter.

Dans le chapitre 2, nous décrivons les approches qui proposent des solutions au problème de la conception avion, et présentons des méthodes de résolutions de problèmes distribués à base d'agents. Ces approches favorisent la description d'un problème par une formulation mathématique pour pouvoir utiliser des approches d'optimisation. Mais cette manière déterministe d'aborder le problème s'explique par un manque d'outils et d'approches permettant de modéliser certaines des caractéristiques de la conception avion, telles que l'interdisciplinarité, la dynamique et l'évolution du problème. Après l'identification des limites dues à ces approches, nous introduisons notre démarche.

Le chapitre 3 présente le contexte théorique qui soutient notre approche. Après avoir défini la notion de systèmes complexes, nous montrons comment les systèmes multi-agents adaptatifs et plus particulièrement la théorie des AMAS (Adaptive Multi-Agent Systems) offrent un cadre théorique pour modéliser et simuler la conception préliminaire avion.

Cette partie se termine par une description des objectifs industriels et scientifiques de la thèse (figure 4).

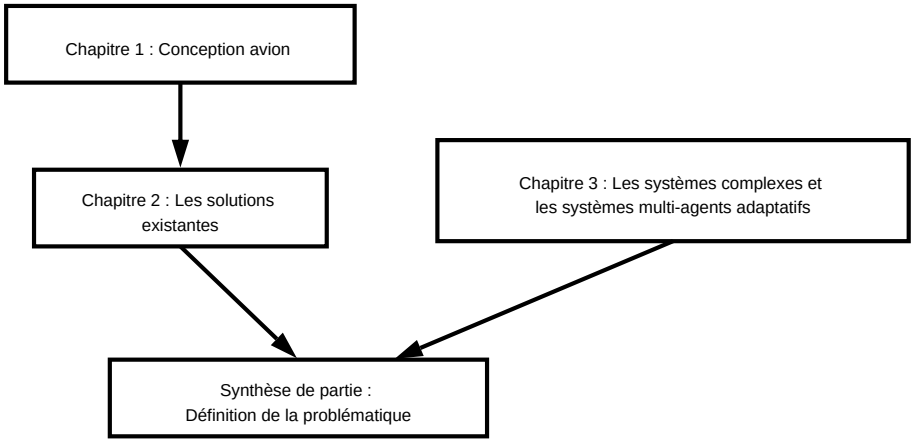


FIG. 2 – Organisation des chapitres de la première partie

Le secret de l'industrie moderne, c'est l'utilisation intelligente des résidus.

Roy Lewis

1

La conception avion et ses besoins fonctionnels

Sommaire

L'avion est un produit complexe, dont les caractéristiques doivent être maîtrisées tout au long du cycle de vie. La complexité du produit est due à de nombreux facteurs tels que :

- le besoin d'interactions entre les phases amonts et avales du projet ;
- le nombre important de composants et de systèmes ;
- l'interdépendance et le nombre d'objectifs visés ;
- les contraintes légales ;

– l'évolution et l'instabilité des spécifications.

Tous ces aspects font de la conception avion un problème complexe, composé de nombreux objectifs et d'enjeux industriels importants.

1.1 Les aspects multi-objectifs et enjeux industriels

Concevoir un avion, c'est donc faire un ensemble de compromis sur ses caractéristiques. En organisant les objectifs de conception, on obtient deux catégories principales : les objectifs provenant des compagnies aériennes et ceux provenant de l'industrialisation.

1.1.1 Les objectifs des compagnies aériennes

Généralement, les objectifs de haut niveau des compagnies aériennes sont de réaliser des missions (transporter un nombre donné de passagers, sur une distance donnée) en ayant des coûts de maintenance et d'exploitation minimaux (DOC *Direct Operating Cost*). Le marché est donc segmenté en fonction de la distance à parcourir et du nombre de passager transporté, figure 1.1. D'autre part, la santé du transport aérien est fortement liée au contexte politique/économique/sécuritaire dans le monde, qui influence la perception que les clients et les autorités ont du risque dans le transport aérien. Pour une compagnie aérienne, ces conditions défavorables peuvent se traduire par des retards, des annulations, des modifications des routes habituelles, etc. Les incertitudes du secteur poussent donc les compagnies aériennes à demander aussi des avions génériques qui puissent évoluer et satisfaire de nouveaux besoins dus aux évolutions rapides et parfois imprévisibles du marché.

1.1.2 Les objectifs industriels

L'objectif industriel de haut-niveau, c'est évidemment de fabriquer des produits en répondant aux demandes des compagnies aériennes et de dégager un maximum de bénéfices. De cet objectif haut niveau découle de nombreux autres.

Les familles d'avions

Aujourd'hui les besoins des compagnies aériennes en rayon d'action sont variés. Lors des dernières années, la prédominance de Boeing sur le secteur des vols longs courriers et gros porteurs (B747) a montré qu'un constructeur aéronautique doit pouvoir proposer une gamme de produit complète aux clients et couvrir au mieux l'ensemble de leur besoin pour peser pleinement dans la compétition internationale. Ce contexte a conduit les

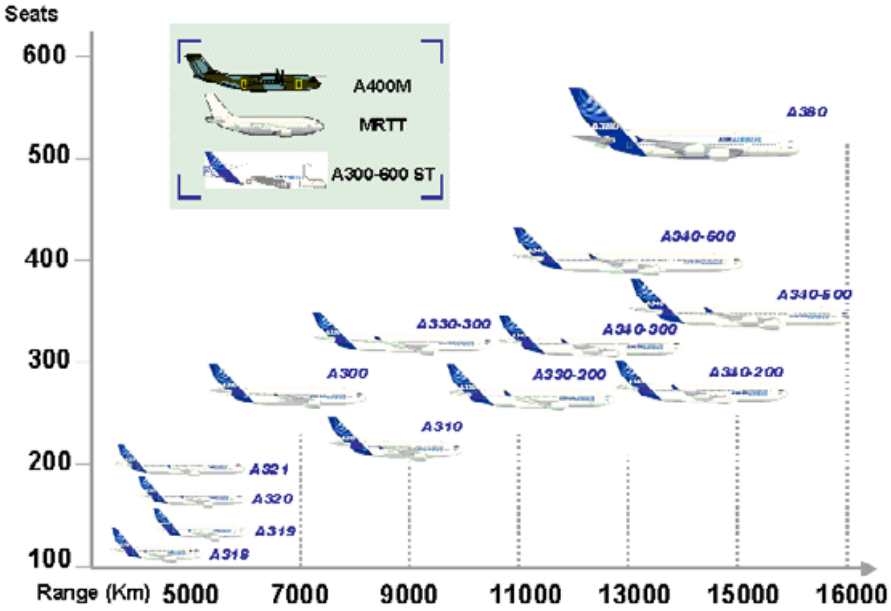


FIG. 1.1 – Famille d'avions

constructeurs à développer une gamme d'avion aux caractéristiques positionnées dans des familles, pour profiter des bénéfices suivants :

- rationaliser l'offre en proposant une gamme de produits ;
- factoriser les efforts de développement et de fabrication, en favorisant la réutilisation de composants ;
- simplifier la chaîne logistique et la politique d'approvisionnement ;
- gérer de manière cohérente les centres de compétences (bureaux d'études et unités de production, etc.).

Cette modularité offre également des avantages aux compagnies aériennes, en leur apportant une réduction des temps de formation et des coûts de maintenance. Concevoir des familles d'avions implique nécessairement de nouvelles contraintes de conception, qui interfèrent parfois avec certains critères de performance et donc avec les exigences de certaines compagnies aériennes, qui veulent des avions aux performances optimales sur chacune de leurs missions. A l'inverse, fabriquer des avions génériques (objectif de familles) implique aussi que le modèle conçu s'écarte légèrement de sa configuration optimale, puisqu'il partage des caractéristiques avec des avions dont les missions sont différentes. Finalement un objectif commun aux compagnies aériennes et aux constructeurs est de mettre sur le marché les avions le plus rapidement possible pour s'adapter à la dynamique du trafic aérien.

Dans ce contexte, les missions des avant-projets sont essentielles. Il s'agit d'une part d'améliorer la vision globale de la famille à concevoir (de prévoir/anticiper les difficultés,

de préciser les phases de conception en amont pour limiter les surcoûts des modifications, etc.), et d'autre part de proposer des solutions innovantes et réalistes aux compagnies aériennes.

1.2 Les aspects multidisciplinaires et le rôle de la conception préliminaire

La conception avion n'est pas seulement un problème d'optimisation multi-objectif, c'est aussi un problème multidisciplinaire, impliquant de nombreux acteurs aux compétences différentes. Parmi ces disciplines, on trouve par exemple : l'aérodynamique, les structures, les commandes de vols, l'estimation de la masse, les motoristes, etc.

1.2.1 La multidisciplinarité

Pour l'ensemble de ces disciplines, l'objectif global est de collaborer pour concevoir un avion réalisant les besoins exprimés par l'avionneur et les compagnies aériennes. Cependant des compromis doivent souvent être faits sur certains objectifs au profit de certaines disciplines. Par exemple, l'avionneur veut vendre à la fois des avions qui soient moins coûteux, plus rapides, plus confortables, plus légers, etc. ; mais l'avion le moins coûteux ne sera certainement pas le plus rapide ni le plus confortable [5] (figure 1.2). Pour trouver des configurations avions intéressantes, il faut donc analyser/affiner le problème multi-objectif de départ, en utilisant les compromis interdisciplinaires qui semblent les plus favorables (ressources humaines disponibles, maîtrise des technologiques envisageables, etc.).

De ce fait, durant cette phase de conception préliminaire, un concepteur se forge une idée globale des compromis interdisciplinaires et affine son problème d'optimisation. Pour cela, il prend en compte les objectifs initiaux (*requirements*) ainsi que les contraintes appliquées aux composants et disciplines, afin de favoriser un équilibre des contraintes. L'affinage du problème se fait donc par une amélioration de sa compréhension, et par une succession de prises de décision.

1.2.2 Les aspects multi-niveaux

Pour maîtriser la complexité du produit, la phase de conception est décomposée en plusieurs jalons [5] (*Milestones*), au cours desquels la conception est décomposée et détaillée. Elle débute après la clôture de la phase de faisabilité et comprend trois phases importantes ; la phase d'optimisation des concepts (M3-M5), la phase de définition (M5-M7) et la phase de développement (M7-M14).

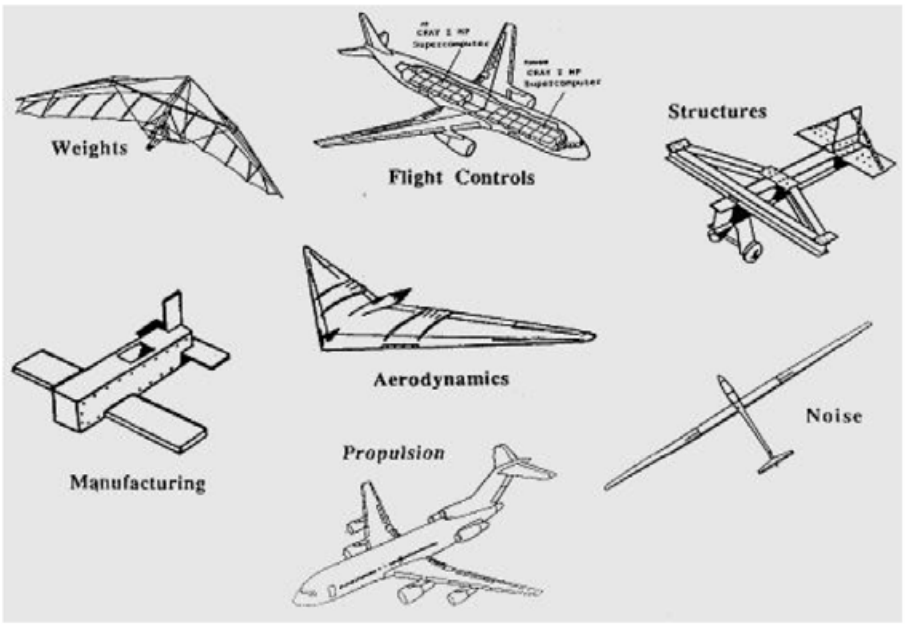


FIG. 1.2 – Caricature des configurations optimales d'avions par discipline

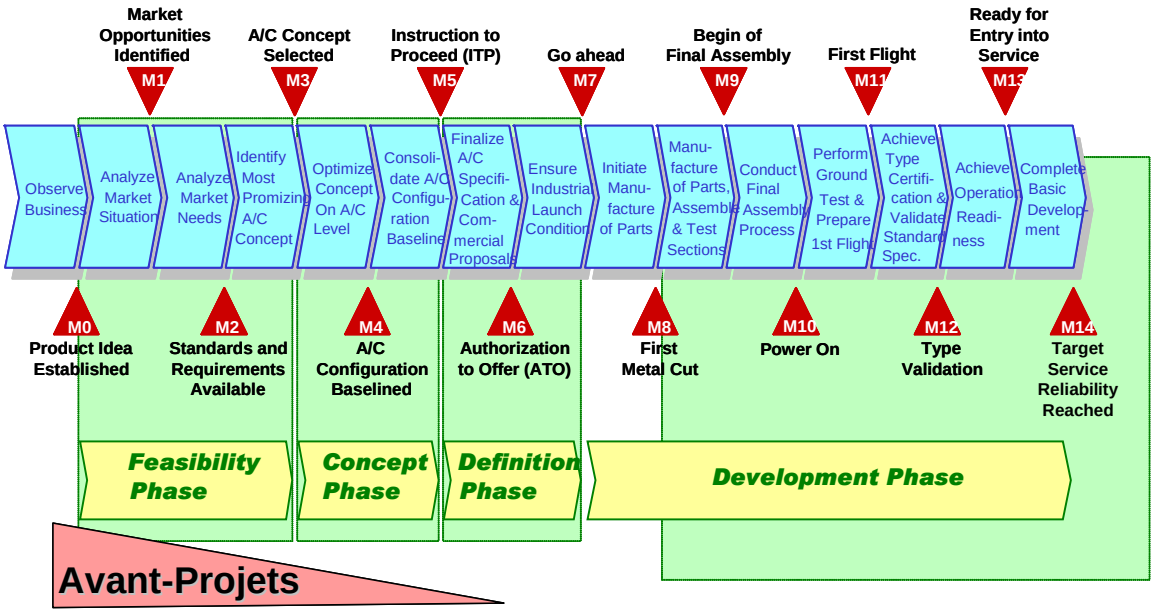


FIG. 1.3 – Phases de la conception avion chez Airbus

1. Durant les jalons M3-M4, on optimise et affine les concepts définis pour l'avion global afin de faire émerger une première configuration, c'est typiquement la phase dédiée aux avant-projets. Après cette activité, une première configuration du produit est proposée (Product Definition Level 0). Les jalons M4-M5 consolident la configuration de référence, en détaillant les concepts techniques associés à l'aide d'outils de conception plus précis et de connaissances métiers approfondies. De cette étape ressort un partage des tâches, un planning du projet, et une configuration de référence plus précise (Product Definition Level 1).
2. La phase de définition débute après le jalon M5. Durant cette phase, le produit passe du stade de concepts détaillés au stade de produit complètement défini par des plans et des modèles. À partir de cette étape, le rôle des avant-projets devient plus consultatif, l'objectif maintenant étant de détailler la conception par composants en utilisant les résultats des étapes précédentes.
3. Finalement, la phase de développement (M7-M14) correspond à la production, l'assemblage et aux tests du premier appareil. Elle permet de préparer la production en série du nouvel avion.

Durant ce découpage en plusieurs niveaux de conception, des choix sont opérés à chaque passage d'un niveau à un autre. Ainsi la définition plus détaillée du produit met l'accent sur de nouvelles contraintes qui nécessitent la recherche de nouvelles solutions plus précises. Mais lorsque les niveaux de précision deviennent plus détaillés, les relations entre les disciplines sont souvent moins évidentes. Ainsi des re-négociations à des niveaux plus abstraits sont alors nécessaires afin d'améliorer la vue d'ensemble et les relations entre disciplines. Ce genre de négociations s'effectue lors des différentes réunions de revue d'un programme avion.

1.2.3 Les enjeux de la conception préliminaire

Les avant-projets jouent un rôle très important dans la conception avion, rôle de définition/conception et de coordination/coopération entre les disciplines et étapes de conception. Les besoins portants sur la conception avant-projet sont donc nombreux et d'autant plus importants que l'on s'intéresse aux phases préliminaires, car les degrés de liberté y sont grands. Les facteurs à prendre en considérations sont très hétérogènes, puisqu'à la fois humains, physiques, technologiques et environnementaux, et sont souvent en contradiction les uns avec les autres. Un ingénieur avant-projet pourra être confronté à des choix aussi divers que :

– évaluer l'impact du choix d'une nouvelle technologie sur la conception, les gains en performances, les coûts en maintenance, etc. ;

– faire des compromis entre les qualités aérodynamiques et l'efficacité des moteurs,
– trouver des caractéristiques géométriques qui facilitent la fabrication de l'avion en l'intégrant dans une famille de produits ;
– évaluer les risques d'une nouvelle technologie.

Dans ce contexte, les sources de complexité sont nombreuses et omniprésentes, car il faut considérer beaucoup de paramètres, maîtriser plusieurs disciplines et intégrer de nombreux résultats. De ce fait l'expérience des concepteurs y joue un rôle important.

1.2.4 Les caractéristiques de la conception préliminaire avion

Un problème inverse

La conception préliminaire avion peut être vue comme un problème d'optimisation, dont les éléments connus sont :

– les caractéristiques fonctionnelles du produit, appelées TLARs *Top Level Aircraft Requirements* (cf. figure 1.4, paramètres Z) ;
– des contraintes sur les paramètres de conception (*design*) de l'avion (X) ;
– la fonction (F) est multidisciplinaire. Elle permet de calculer les caractéristiques fonctionnelles (Z) à partir d'un ensemble de paramètres de conception (X). Cette fonction est construite par l'assemblage de modèles disciplinaires, où chaque modèle représente une certaine physique ou un métier avion. Par exemple, notre modèle de *Mission* permet de calculer le *rayon d'action* de l'avion en utilisant la relation de Breguet-Leduc. Cette relation (équation 1.1) est exprimée en fonction de la finesse de l'avion (sa performance aérodynamique), de sa consommation, de sa vitesse et du rapport entre sa masse à vide et sa masse au décollage.

$$Range = \frac{Finesse\ de\ l'avion}{Consommation} * Log\frac{Masse\ au\ décollage}{Masse\ à\ vide} * Vitesse \tag{1.1}$$

En utilisant ces données, l'objectif est de trouver un jeu de paramètres X qui permette d'obtenir les performances Z en utilisant la fonction de calcul F. Mais les éléments connus étant F et Z, il est nécessaire d'itérer le processus de simulation sur des jeux de paramètres différents afin de converger vers un ensemble de paramètres X qui soit satisfaisant. Le processus itératif est présenté en figure 1.4 et peut être résumé de la façon suivante :

1. formuler les caractéristiques fonctionnelles Z et la fonction de simulation F ;
2. itérer un processus de simulation sur un ensemble de valeurs X, tel que Z' = F(X) ;
3. comparer les paramètres Z et Z' ;
4. modifier les paramètres X (jusqu'à ce que Z et Z' soient proches).

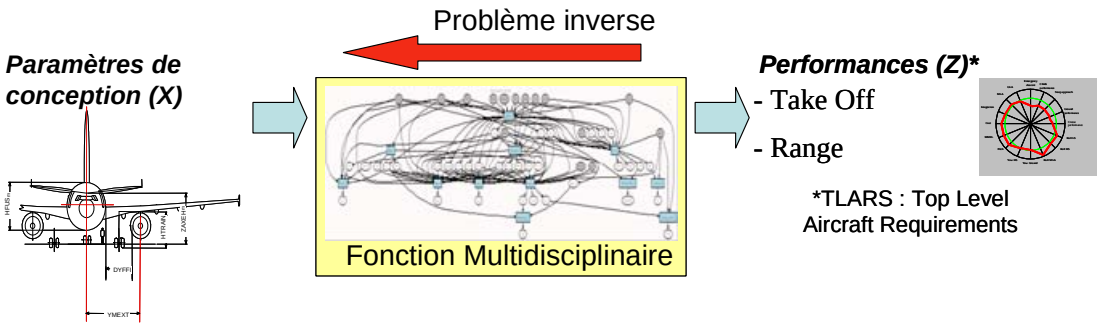


FIG. 1.4 – Le problème inverse de la conception avion

Le bouclage *Masse/Performance*

La fonction de simulation de la figure 1.4 est multidisciplinaire, car elle modélise des processus métiers. Une de ses particularités est qu'elle est composée de boucles de rétroactions, modélisant les contraintes imposées par le processus *Masse/Performance*. Ce processus de la responsabilité des avant-projets sert à définir la masse maximale au décollage de l'avion en fonction de la charge utile et du rayon action. Ce processus est fortement interdépendant puisqu'une mission, pour être réalisée, nécessite une quantité de carburant, qui se répercute sur la masse maximale au décollage, sur les structures de l'avion et donc sur la mission.
Au niveau de la fonction de simulation, ces contraintes se traduisent par un besoin pour le modèle d'évaluation de la masse de connaître le rayon d'action, et inversement pour le modèle d'évaluation de la mission de connaître la masse de l'avion. De ce fait, la masse au décollage et la mission sont à la fois des hypothèses de calcul à fournir en entrée de la fonction et des résultats d'évaluation. Par conséquent, le système est cohérent lorsque les valeurs de la masse et du rayon d'action fournies en entrée sont égales aux valeurs obtenues en sortie.
Concrètement et tel que l'illustre la figure 1.5, ce problème est résolu par une itération du système sur la masse au décollage et sur la mission. Une solution à ce problème de *Masse/Performance* est trouvée lorsque les deux contraintes opérationnelles d'égalité sont satisfaites simultanément.

1.2.5 Les étapes de la conception préliminaire

Pour réaliser un nouvel avion, une étude avant-projet est organisée en plusieurs étapes. Les figures 1.6 et 1.7 résument les principales étapes et les différents besoins fonctionnels d'une conception avant-projet :

1. Dans un premier temps, une ébauche des besoins est réalisée. Elle permet, en ana-

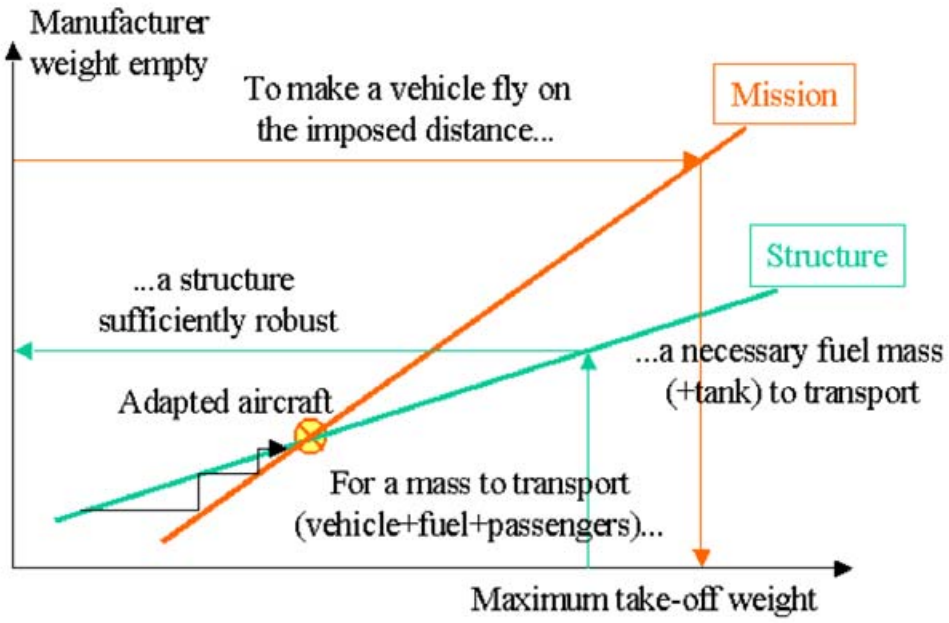


FIG. 1.5 – Le bouclage Masse/Performance [5]

lysant les attentes commerciales et les évolutions du marché, de proposer une série de besoins et d'objectifs. Une fois cette série d'objectifs définie, on s'appuie sur les expériences passées et sur les avions connus pour trouver, parmi ceux existants, celui qui est le plus proche de l'étude, et que l'on l'appelle avion de référence.

2. Les objectifs de la première étape définissent le contexte de l'étude, et permettent de sélectionner un avion le plus proche de ce contexte. Durant la seconde phase, le concepteur collecte des briques de modélisation (de modèles) qui sont utilisables dans le contexte de l'étude. Cet ensemble de briques comprend des modèles d'aérodynamique, de calcul de masse, etc.
3. Avec l'ensemble des modèles de simulation sélectionnés, le concepteur construit une fonction de simulation et lui applique l'avion de référence, pour lequel les performances réelles sont connues. En comparant les performances fournies par les modèles choisis avec les performances réelles de l'avion, on vérifie la qualité des résultats fournis par l'assemblage de modèle. Ainsi lorsque les résultats fournis par les modèles ne sont pas satisfaisants, on les ajuste en utilisant des outils de calibrage.
4. À ce stade de la modélisation de nombreux degrés de liberté (paramètres de conception) sont disponibles. Afin de définir un problème optimisable, le concepteur sélectionne des topologies géométriques qui permettent de réduire le nombre de paramètres. Par exemple, les caractéristiques d'une aile seront modifiées à l'aide de quelques paramètres.

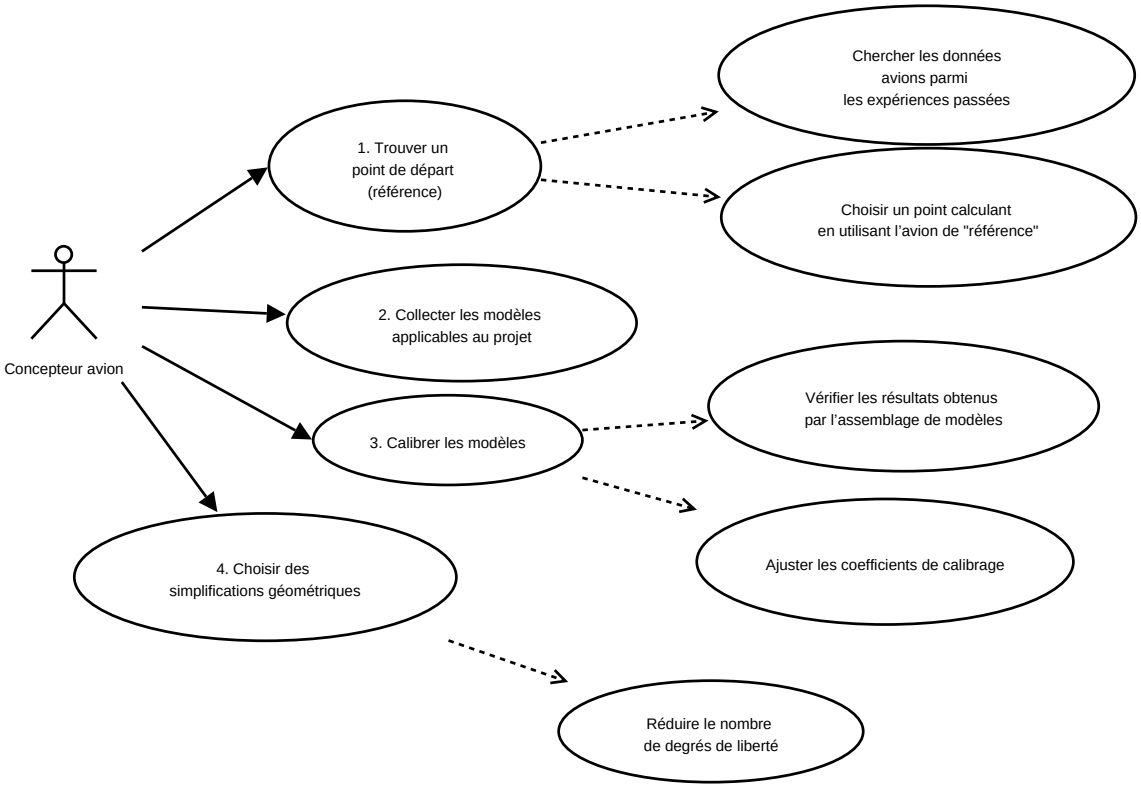


FIG. 1.6 – Formulation du problème et définition des éléments de simulation

5. Une fois que les modèles sont calibrés et que les degrés de liberté sont définis, le concepteur introduit de nouveaux coefficients de calibrage, qui vont permettre de tenir compte des innovations technologiques disponibles ou envisagées. La fonction est alors utilisée pour trouver un ensemble de paramètres de conception qui satisfont le nouveau cahier des charges. Cependant pour simplifier le problème et la recherche de ce point de départ, l'étude est souvent réalisée dans un premier temps sans étudier le bouclage *Masse/Performance* (section 1.2.4). À ce stade, la conception est donc incomplète et non cohérente, puisque la masse fournie en entrée et en sortie du système n'est pas équivalente. Mais ce point de départ est qualifié de *calculant*, puisqu'il respecte les contraintes (performances) du problème.

6. Ensuite, le nouvel avion est étudié avec une chaîne de conception complète, qui inclut un bouclage *Masse/Performance* cohérent. La recherche de la cohérence est lancée à cette étape. Mais parfois le bouclage n'est pas réalisable avec le paramétrage obtenu à l'étape précédente et certains objectifs doivent alors être modifiés. Dans ce cas, il est au moins nécessaire de recommencer l'étape précédente.

7. Lorsque la chaîne de conception est vérifiée, une première optimisation du nouvel avion peut être engagée. Elle débute par la sélection des critères à optimiser. Des

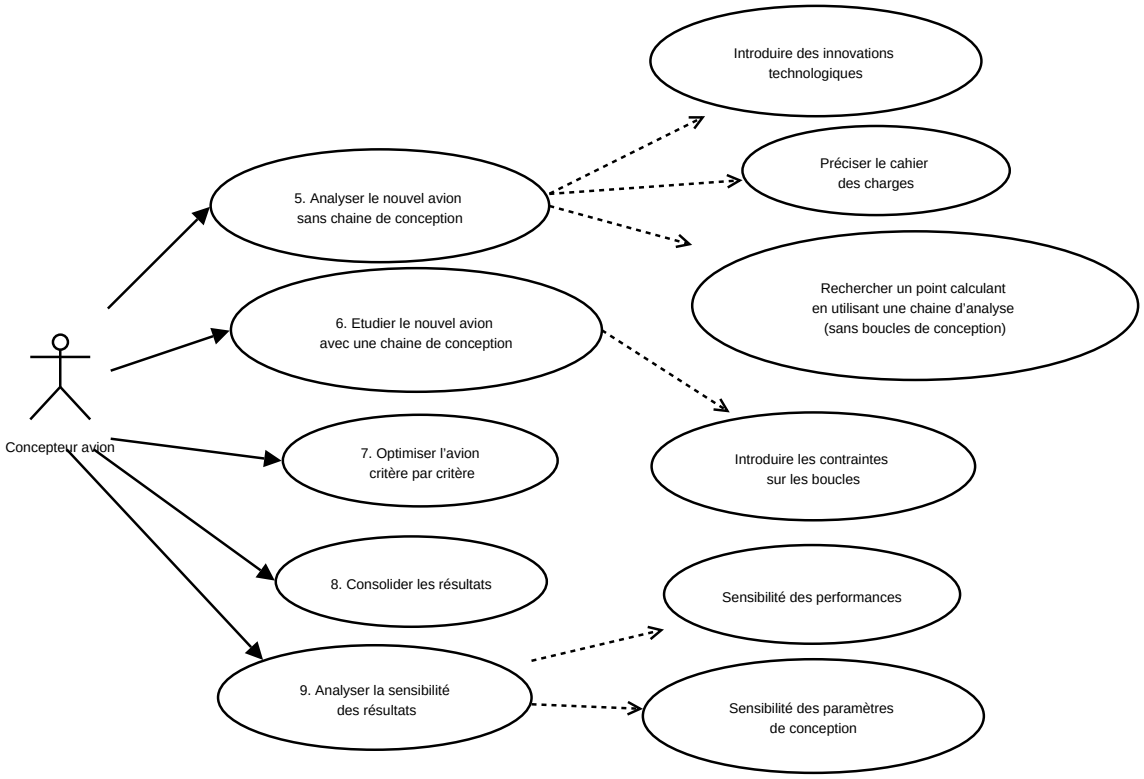


FIG. 1.7 – Construction de configurations avions de haut niveau

optimisations critère par critère sont ensuite réalisées afin de proposer les configurations optimales pour chacun d'entre eux.

8. Une fois ces résultats obtenus, une phase d'analyse et de consolidation est réalisée. Elle consiste à confronter les résultats à d'autres sources de calcul ou à d'autres experts du domaine, jusqu'à présent extérieurs à l'étude. On teste alors différentes hypothèses et on vérifie si d'autres choix plus pertinents n'auraient pas été possibles.

9. Finalement, la dernière étape consiste à sélectionner une solution et une configuration et à lui appliquer des analyses de sensibilités. L'objectif est alors de vérifier que la solution trouvée n'est pas trop sensible aux modifications des paramètres de conception ou des performances.

Par souci de simplicité, les différentes étapes présentées de façon séquentielle dans la réalité de nombreuses boucles sont possibles et l'avancement de l'étude peut être différent en fonction de la discipline.

1.3 Les besoins fonctionnels d'une application aidant à la conception avion

Ces besoins fonctionnels de haut niveau nécessitent la mise en place de nouveaux outils informatiques. À titre indicatif, voici une liste d'outils, susceptibles de répondre à certains besoins :

1. outils de sélection de modèle, nécessitant la mise en place de bases de données des modèles ;
2. outils de supervision des modèles lors de leur exécution, afin de vérifier que leur utilisation est conforme à leur spécification ;
3. outils de gestion des configurations, qui en relation avec la base de données de modèles permettent de garder une trace des évolutions et des liens existants entre les modèles (liens de calibrage, d'évolution technologique...) ;
4. outils d'assistance pour trouver des points calculants et redéfinir les contraintes ;
5. outils d'aide à l'optimisation, pour affiner les choix des critères à optimiser et percevoir des gains concurrentiels ;
6. outils d'analyse de sensibilité facilitant la consolidation des résultats et limitant les risques.

Certains de ces outils peuvent être associés directement à une étape de la conception préliminaire avion. Cependant, certaines étapes de cette décomposition de la conception correspondent à une simplification du problème qui permet de procéder étape par étape. Par exemple, les étapes d'optimisation et d'analyse de sensibilité sont forcément liées, puisqu'une solution optimisée mais sensible et moins intéressante qu'une solution optimisée et robuste aux changements. L'analyse de sensibilité devrait donc être menée simultanément et non *a posteriori*.

1.4 Objectif de la thèse

Nous proposons de favoriser la prise en compte du problème dans sa globalité en le modélisant avec un système multi-agent (SMA), composés d'éléments logiciels autonomes (agents). Ils offrent un cadre conceptuel permettant la représentation et la simulation de systèmes complexes grâce à leurs capacités :

– de représentation des connaissances en local, chaque discipline impliquée dans la conception avion peut être modélisée de manière indépendante à l'aide d'un agent ;

- d'interaction/négociation, en utilisant les interactions et les capacités de coopération des agents, on peut réaliser des compromis interdisciplinaires et en déduire des propriétés fonctionnelles et globales de l'avion, qui ne peuvent être facilement comprises en examinant simplement et isolément les besoins de chaque modèle disciplinaire ;
- de distribution, chaque modèle disciplinaire peut être distribué ;
- d'évolution, chaque modèle disciplinaire a son propre cycle de vie sans que cela n'ait un impact sur la modélisation du système global ;
- d'ouverture, de nouveaux modèles peuvent apparaître et disparaître sans que cela ne remette en cause le fonctionnement global du système.

De ces propriétés découlent de nouvelles perspectives en terme de compréhension globale, d'intégration de disparités interdisciplinaires et d'étude des conséquences d'un comportement individuel sur la globalité du système, etc.

L'objectif de cette thèse est de proposer une nouvelle approche d'aide à la conception avion qui réponde aux besoins des étapes 5, 6, 7 et 8, en utilisant les capacités de coopération d'un système multi-agent adaptatif. Dans ce système, les agents représentent les modèles disciplinaires, les paramètres de conception et les performances de l'avion. En utilisant des connaissances disciplinaires et par un comportement coopératif, ces agents trouvent collectivement les valeurs des paramètres de conception qui satisfont les contraintes et les performances.

Les propositions mathématiques sont reçues comme vraies parce que personne n'a intérêt qu'elles soient fausses.

Montesquieu

2

L'analyse des solutions existantes

Sommaire

Tel que décrit précédemment, la conception avion est un problème inverse difficile à résoudre car composé d'interdépendances fortes. Actuellement, il existe deux approches pour traiter ce type de problèmes.

La première approche l'aborde d'une manière globale, en englobant l'ensemble des données et des modèles dans une unique fonction. Des techniques d'optimisation multi-objectif servent ensuite à résoudre les conflits.
À l'inverse, la seconde approche cherche à utiliser la structure du problème pour le décomposer en plusieurs sous-problèmes. Dans cette seconde approche, on cherche des solutions aux conflits en utilisant les relations entre les modèles.

Dans cette partie, nous donnerons un aperçu des techniques d'optimisation multi-objectif (*Multi-Objective Optimisation* MOO), qui répondent très bien à la première approche. Nous verrons ensuite comment certaines techniques d'optimisation distribuées tentent de répondre à la seconde, en particulier nous introduirons principales approches d'optimisation multi-disciplinaire (*Multi-Disciplany Optimisation* MDO) ainsi que d'optimisation combinatoire à base d'agents.

2.1 L'optimisation multi-objectif (*MOO*)

En traitant le problème de la conception avion de manière globale, on peut définir un ensemble d'entrées/sorties, pour lesquels on se fixe des objectifs, et une fonction d'évaluation caractérisant le problème. Ainsi on peut utiliser l'optimisation multi-objectif pour traiter ce problème.

2.1.1 La formulation du problème

Un problème multi-objectif peut être défini comme un problème, dont on recherche l'état qui satisfait un ensemble de contraintes et optimise un ensemble de fonctions objectifs. Les contraintes s'appliquent sur les paramètres pour lesquels on a des exigences à respecter. Les fonctions d'objectif définissent des fonctions de coût à minimiser qui permettent de prendre en compte les préférences que l'on peut avoir sur certains paramètres. Prenons l'exemple du paramètre *Rayon d'action* d'un avion, la contrainte assure un rayon minimal à réaliser, qui implique que pour chaque solution au problème sa valeur devra être supérieure à celle imposée par la contrainte. Alors que la fonction objectif associée à ce même paramètre a pour rôle de favoriser la recherche d'une solution qui maximise la valeur du *Rayon d'action*. En général, les fonctions d'objectif sont interdépendantes les unes avec les autres et ces problèmes d'optimisation ont plusieurs solutions, puisque minimiser une des fonctions en augmente souvent une autre. La notion d'optimum ne peut donc pas être clairement établie.

De ce fait, un problème multi-objectif se définit de la façon suivante :

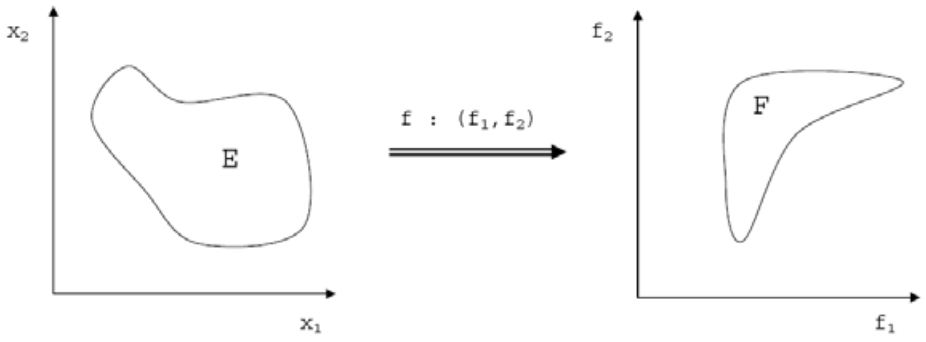


FIG. 2.1 – Projection de l'espace des paramètres vers l'espace des objectifs

– **un vecteur de décisions**, composé des variables du problème :

$$x = (x_1, x_2, ...x_n) \tag{2.1}$$

avec n le nombre de variables

– **un ensemble de contraintes**, notées :

$$g_i(x) \text{ avec } i = 1, .., m \tag{2.2}$$

avec m le nombre de contraintes.

– **un vecteur de fonctions objectif** noté f :

$$f(x) = (f_1(x), f_2(x), ..., f_k(x)) \tag{2.3}$$

avec f_i les objectifs à optimiser et k le nombre d'objectifs. Traditionnellement les fonctions objectifs f_i sont écrites de telle sorte qu'elles sont à minimiser.

Un problème d'optimisation recherche donc un vecteur x, qui satisfait l'ensemble des contraintes et optimise les fonctions portant sur les objectifs. Ainsi les domaines de définition de chaque variable et l'ensemble des contraintes permettent de définir un espace d'actions réalisables, qui se projette dans l'espace des objectifs souhaités, comme l'illustre la figure 2.1. Une des difficultés des problèmes multi-objectifs est qu'il n'existe pas de définition de la solution optimale. En général, le décideur peut seulement exprimer le fait qu'une solution est préférable à une autre, mais il n'existe pas une solution meilleure que toutes les autres.

2.1.2 La recherche et la comparaison des solutions

Pour répondre à ce type de problème, la communauté scientifique a proposé principalement deux approches. La première consiste à ramener le problème à une optimisation mono-critère, alors que la seconde prend simultanément en compte l'ensemble des critères.

L'agrégation des objectifs

Pour ramener un problème multi-objectif à un problème mono-objectif, il suffit d'utiliser des méthodes d'agrégation des critères. Plusieurs méthodes ont été proposées : moyenne pondérée, goal programming ou min-max [10].
Mais ramener le problème à du mono-critère présente des limites évidentes. Tel que décrit précédemment, un problème d'optimisation multi-objectif est caractérisé par des inter-dépendances qui font qu'une solution optimale, pour un objectif donné, ne correspond pas à celles des autres objectifs pris indépendamment. La plupart du temps, il n'existe donc aucun point de l'espace de recherche pour lequel toutes les fonctions objectifs soient optimales simultanément. Les décideurs sont donc à la recherche de solutions alternatives, ou en tous cas d'indication sur les caractéristiques d'un résultat.
De ce fait, la transformation du problème en un problème mono-objectif est délicate car :

- Si l'optimisation mono-objectif peut garantir l'optimalité de la solution, elle en trouve une seule. Elle est donc non adaptée à la présentation/recherche d'alternatives de conceptions chères aux décideurs. Ainsi pour différentes situations, le problème doit être résolu plusieurs fois.
- La définition des objectifs de l'utilisateur est toujours bruitée. En général, il lui est difficile de définir précisément et dès le début ses critères de recherche. La mono-criticité est donc sensible à l'ensemble des paramétrages (définition des contraintes, passage à une fonction de coût unifiée, etc.).

Finalement, le but d'un problème multi-objectif est de trouver de "bons compromis" plutôt qu'une seule solution. Lorsqu'il y a plusieurs objectifs, la notion d'optimum change, il est alors préférable d'utiliser un autre terme, le plus couramment adopté étant *l'optimum de Pareto*.

L'approche Pareto

L'approche Pareto est issue des travaux en économie de Edgeworth et Pareto [62]. Elle est basée sur la notion de dominance dans la sélection des solutions générées. Contrairement aux approches qui utilisent une fonction d'utilité globale, elle fournit un ensemble de solutions à un problème sous la forme d'un front de Pareto.

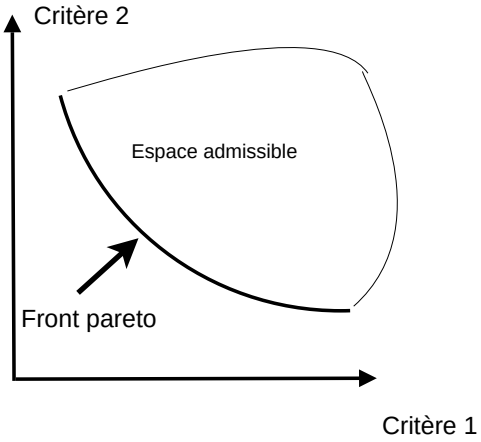


FIG. 2.2 – Exemple de front de Pareto pour deux fonctions objectif

Une solution de Pareto Considérons un ensemble d'objectifs, une solution est dite *Pareto optimale* sur cet ensemble de paramètres si aucune amélioration sur un des objectifs n'est possible, sans la dégradation d'au moins un des autres objectifs.
Pour $f(x) = (f_1(x), f_2(x), ..., f_n(x))$ l'ensemble des fonctions objectifs à minimiser, un vecteur de variables $x^* \in S$ (S région réalisable) est une solution dominante par rapport à un autre vecteur solution $x \in S$, si $\forall j f_j(x^*) \leq f_j(x)$ et $\exists i$ tel que : $f_i(x^*) < f_i(x)$

Les fronts de Pareto

En utilisant la définition précédente et en l'appliquant à un problème d'optimisation, on obtient un ensemble de solutions dominantes. En représentant graphiquement ces solutions, on peut alors mesurer leur niveau d'interdépendance.

La figure 2.2 illustre un front de Pareto pour une fonction objectif f à deux dimensions. Pour représenter graphiquement le front, on associe chaque dimension à un des axes de la figure 2.2. Ainsi la courbe représente l'ensemble P des solutions *Pareto optimales*, qui sont les meilleurs compromis possibles au problème. En se déplaçant sur la courbe marquée en gras, on choisit selon la direction d'améliorer un critère ou l'autre ; l'amélioration de l'un se faisant forcément au détriment de l'autre.

L'utilisation de la notion de rang pour construire le front

Pour construire le front de Pareto, on utilise simplement la définition de la notion de dominance. Considérons à nouveau un problème d'optimisation à deux critères, la figure 2.3 présente, dans l'espace des objectifs, un ensemble de solutions, duquel il faut extraire les solutions Pareto optimales.

Pour établir les relations de dominance entre ces diverses configurations, un certain

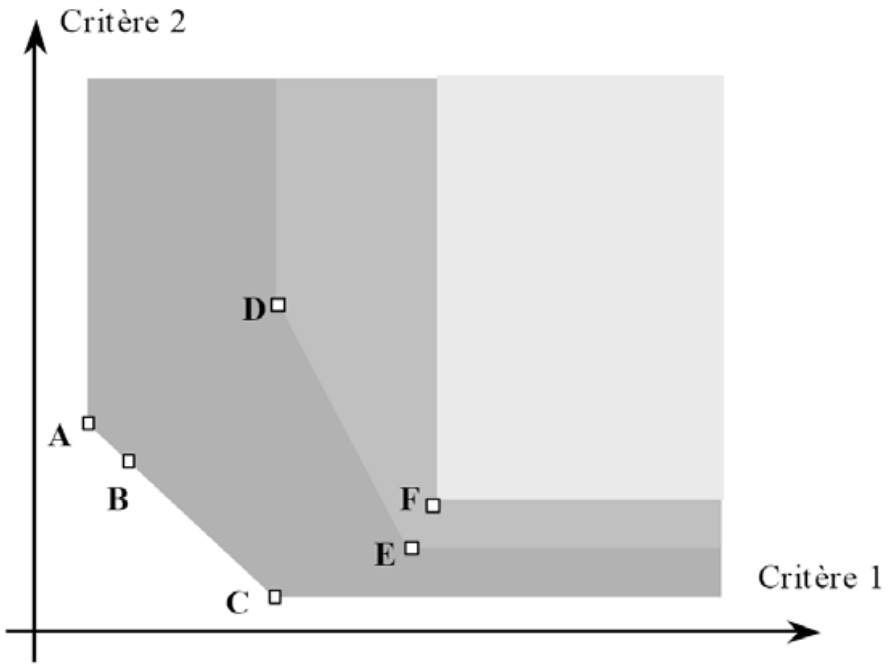


FIG. 2.3 – Exemple de la notion de dominance

nombre de comparaisons est nécessaire. Chacune des solutions examinées doit être comparée à toutes les autres. En effet, dire par exemple que A n'est pas dominée par B ne signifie en aucun cas que A domine B.

En analysant la figure 2.3, on en déduit que le front de Pareto est constitué des solutions non dominées A, B, et C et que : A domine D ; B domine D ; C domine D, E et F ; E domine F ; D et F ne dominent aucun point.

La comparaison des différentes solutions permet alors de définir la notion de rang :

– une solution qui n'est pas dominée est dite de rang 0 ;

– une solution qui est dominée par une solution de rang n est dite de rang $n + 1$.

Ainsi A, B et C sont de rang 0 ; D et E sont de rang 1 ; enfin F est de rang 2.

La comparaison des solutions à l'aide des fronts de Pareto présente de nombreux avantages, et est souvent bien plus efficace que la simple agrégation de paramètres. Elle est néanmoins vouée à certaines limites, notamment lorsque la dimension du problème augmente (taille des vecteurs de décision et d'objectifs), car dans ce cas, la représentation graphique ainsi que le parcours des fronts deviennent difficiles.

2.1.3 Les méthodes de résolution

Différentes méthodes d'optimisation s'appliquent selon le mode de représentation que l'on choisit (Pareto ou non Pareto) :

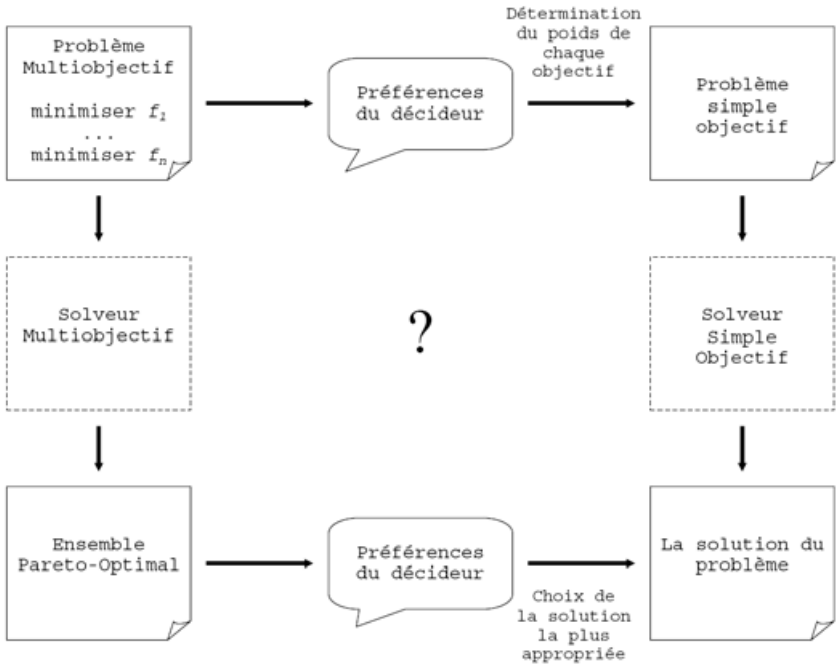


FIG. 2.4 – Décision *a posteriori* vs. *a priori* sur la formulation du problème

- Dans le cas où l'on revient à un problème mono-objectif, le décideur/concepteur intervient en amont du processus d'optimisation. La solution que le système lui présente correspond aux préférences qu'il a établies lors du choix de sa méthode d'agrégation de critères. Ces approches sont dites *a priori*.
- A l'inverse dans les approches *a posteriori*, le décideur intervient en aval du processus d'optimisation. Dans ce cas, il exerce son choix final parmi un ensemble de solutions non dominées. La figure 2.4 illustre les interventions de l'utilisateur avec ces deux modes de résolution.
- Il existe une troisième approche dite *interactive* qui permet au décideur de définir des préférences au cours de la résolution du problème. Dans ce cas, la formulation du problème évolue et donc les solutions dominantes aussi.

Chacune de ces approches génère des besoins différents à la fois en termes de résolution et de représentation des résultats. Elles ont donc conduit à la proposition d'algorithmes assez différents.

La résolution par décision *a priori*

Les méthodes de résolution *a priori* sont les plus utilisées en milieu industriel. Leur avantage est qu'elles se basent sur des algorithmes bien connus et éprouvés. Mais ces approches sont dites *naïves* [19], car elles ne s'intéressent pas à la nature du problème.

En pondérant et en agrégeant les objectifs, on se prive d'une partie des solutions ainsi que d'une amélioration de la compréhension du problème traité, tel que décrit précédemment 2.1.2.
Mais une fois le problème d'optimisation devenu mono-critère, tous les algorithmes traditionnels d'optimisation sont utilisables, tels que : le gradient, le recuit simulé, la recherche tabou, etc. D'une manière générale et pour des problèmes de conception aux nombreuses interdépendances, les méta-heuristiques basées sur des approches stochastiques (recuit simulé, Monté Carlo...) sont préférées aux approches locales ou à base de gradient, car plus robustes aux discontinuités de l'espace de recherche et aux minima locaux. Pour une description approfondie de ces algorithmes, on pourra se référer à [27].

La résolution par décision *a posteriori*

Le choix d'une approche *a posteriori* implique nécessairement la recherche d'un panel de solutions non dominées au sens de Pareto, parmi lesquelles le concepteur pourra faire ses choix. À nouveau, les méthodes stochastiques ou évolutionnaires sont couramment utilisées et ceci pour les mêmes raisons que précédemment. On retrouve donc essentiellement des méthodes basées sur du recuit simulé [19], des algorithmes de fourmi [1], des algorithmes d'essaims particulaires [14] et des algorithmes génétiques. Néanmoins quelques approches locales à base de recherche tabou ont aussi été expérimentées [19].

Les méthodes traitant ce type de problème se séparent principalement en deux catégories :

- Les méthodes non-Pareto utilisent des opérateurs, qui traitent séparément les différents objectifs. C'est-à-dire qu'elles favorisent la recherche des solutions dominantes pour chaque objectif pris individuellement. Par exemple, des algorithmes génétiques à sélection parallèle permettent de faire évoluer simultanément plusieurs populations, où chacune a des modes de sélection favorisant l'optimisation d'un objectif différent. En favorisant la recherche de la solution qui optimise chaque objectif d'une manière indépendante, et en s'autorisant des échanges d'individu entre population, elles finissent par trouver des bonnes solutions au sens de Pareto.
- À l'inverse, les méthodes dites Pareto recherchent directement des solutions non dominées, en utilisant une mémoire d'individus dominants. Dans ce cas, la recherche des solutions ne favorise pas la recherche d'un critère en particulier, mais celle de meilleurs individus non dominés. Dans ce cas, les solutions sont mémorisées puis réutilisées pour attirer l'ensemble de la population vers les fronts. Mais dans ce type de résolution, l'ajustement des paramètres de l'algorithme génétique est important puisqu'il assure un équilibre entre l'exploration de nouvelles solutions et l'exploita-

tion des solutions identifiées comme dominantes.
Quelques-unes de ces approches seront décrites dans le chapitre 5.

La résolution par des approches décisionnelles *interactives*

Ces approches sont les plus variées, car elles ont été étudiées pour traiter des problèmes, où :

- soit on se ramène à du mono-critère ;
- soit on cherche un panel de solutions multi-objectifs, que l'on adapte.

Dans le premier cas, on utilise simplement des méthodes d'optimisation traditionnelles, qui permettent de trouver un point de départ puis de l'adapter selon les interactions avec le concepteur. Le point de départ est souvent une solution non dominée de Pareto, qui est atteinte en utilisant une première agrégation de paramètres. La logique de ces approches est ensuite d'utiliser des analyses de sensibilité autour de ce premier point pour aider le concepteur à affiner sa recherche, et donc à modifier ses contraintes et/ou les poids de son agrégation de paramètres. En se déplaçant de point en point, le concepteur finit par trouver une solution qui le satisfait. C'est le cas par exemple de l'approche iMOODS [74], présentée en partie 5 et de NIMBUS [50].

Dans le second cas, on utilise plutôt des approches multi-objectifs, inspirées de celles utilisées dans la prise de décision *a posteriori*. Néanmoins une dimension supplémentaire est ajoutée à ces méthodes, qui permettent de garantir plus d'exploration et d'adaptation. D'une manière générale, ceci se traduit par une diversification des individus, capables d'apporter de nouvelles solutions en cas de changements environnementaux. L'objectif est de préserver un vivier d'individus capable d'occuper une grande partie de l'espace de recherche, et d'utiliser cette caractéristique comme un facteur d'adaptation lorsque les objectifs changent. C'est par exemple le cas des approches proposées par [10] ainsi que la plupart des algorithmes basés sur une population d'individus.

2.1.4 Synthèse et limites des approches MOO

Comme nous venons de l'illustrer, les algorithmes multi-objectifs offrent un panel important de solutions, car l'ensemble de ces solutions a été étudié pour répondre à des besoins et des choix conceptuels bien précis. Ainsi lorsque l'on a à traiter un problème d'optimisation multi-objectif, le choix de la démarche doit se faire en essayant de répondre à de nombreuses questions, dont voici une liste non exhaustive :

- **Le problème permet-il et/ou nécessite-t-il la recherche d'un ensemble de solutions ?** Ce critère permet de choisir entre des approches par agrégation de paramètres offrant une unique solution et celles à solutions multiples.

- **Le problème à traiter est-il fortement couplé et/ou composé de minima locaux ?** Ce choix est délicat, car on ne connaît jamais exactement la structure du problème, mais il conduit souvent au choix d'une méthode d'optimisation par recherche locale (gradient, recherche tabou, etc.) ou globale (méthodes stochastiques : algorithmes de fourmis, à particules, évolutionnaires, etc.). Cependant dans le cas où l'on recherche un panel de solutions, on préfère souvent les méthodes stochastiques.
- **La dimension du problème permet-elle de rechercher l'ensemble du front de Pareto ?** Dans ce cas, on doit définir une stratégie de recherche des solutions en fonction du nombre de variables, et qui soit une stratégie Pareto ou non-Pareto.
- **L'expert a t-il une bonne compréhension du problème ?** La réponse à cette question peut aider à définir le rôle de l'utilisateur : en amont, en aval ou durant le processus de résolution.

Toutes ces questions sont loin d'être indépendantes les unes des autres, et le choix de la démarche la plus adaptée à un problème donné est souvent difficile, et de nombreuses améliorations peuvent être apportées. Le passage à l'échelle pose par exemple les problèmes suivants :

La représentation des solutions : dans les approches exploratoires (décision *a posteriori*), la représentation des solutions sous la forme de front de Pareto devient difficile dès que le nombre d'objectifs est supérieur à 3, et donc il en est de même pour leur analyse et leur compréhension. Or on sait que la qualité finale des solutions obtenues dépend avant tout de la compréhension du problème et des choix réalisés par le concepteur, il s'agit donc, à l'heure actuelle, d'une limite. De la même manière, les approches d'agrégation de paramètres (*a priori*) sont également difficilement envisageables, sans que le concepteur n'ait des idées précises sur la nature des solutions.

L'augmentation du nombre de degrés de liberté : elle pose des problèmes aux approches *a posteriori*, basées sur des méthodes stochastiques car elle conduit souvent à une explosion combinatoire ou à une mauvaise exploration de l'espace.

En plus de ces limites, l'augmentation des degrés de libertés pose également de nouvelles questions sur l'impact de la modélisation du problème et sur la qualité des résultats. Ainsi la robustesse des résultats et l'amélioration des compromis disciplinaires sont autant d'éléments importants à prendre en compte dans des problèmes tels que celui de la conception préliminaire avion. Des méthodes d'analyse de sensibilité et de robustesse viennent donc se greffer à l'optimisation multi-objectif.

La robustesse et l'analyse de sensibilité

Bien souvent, la robustesse d'un résultat est vue comme un problème d'incertitudes, qui consiste à mesurer la sensibilité des objectifs par rapport aux paramètres de décision. En conception multi-disciplinaire, les incertitudes liées à une discipline sont propagées à travers les autres disciplines. Ainsi, l'incertitude obtenue sur les paramètres de sortie est due à une propagation d'incertitudes cumulées tout au long de la simulation [28]. Mais dans le cas d'une simulation, les sources d'incertitudes sont nombreuses (incertitudes sur paramètres, incertitudes sur la précision d'un modèle), il s'agit donc d'un problème difficile, qui suscite actuellement un grand intérêt en mathématiques appliquées [5]. Mais d'autres facteurs de robustesse sont souvent moins étudiés dans la littérature et tout aussi importants dans l'analyse de risque d'une configuration avion. En effet, un concepteur cherche une configuration qui satisfasse ses objectifs, qui soit robuste aux incertitudes propagées, mais aussi pour laquelle les interdépendances entre les disciplines ne soient pas trop fortes.
Ce dernier aspect consiste à anticiper les problèmes qui pourraient apparaître lors des phases de conception plus détaillées. Il s'agit de déterminer quels sont les paramètres les plus couplés et d'évaluer les degrés de libertés restant pour les étapes suivantes de la conception. Plusieurs approches ont été proposées pour apporter des solutions au concepteur. Dans [5], une méthode construit des ellipsoïdes à partir des frontières du domaine admissible. Il s'agit d'utiliser les données obtenues après optimisation du problème pour trouver les paramètres de l'ellipsoide au volume maximal qui s'inscrit dans cet espace. Ainsi le paramétrage de l'ellipsoïde établit un niveau de dépendance entre les objectifs.
Cependant cette méthode présente quelques inconvénients, elle suppose que l'ensemble des frontières de l'espace admissible soient connues, que l'espace soit relativement homogène et que le nombre de degrés de liberté ne soit pas trop important. D'autres approches d'apprentissage non supervisées plus classiques, telles que les réseaux de neurones [47] ou des cartes auto-organisatrices [65] pourraient être utilisées pour traiter ce type de problème. Cependant tous ces algorithmes nécessitent des données pour faire leur apprentissage qui présupposent donc que l'on dispose d'une bonne connaissance de l'ensemble des solutions admissibles, autour d'un point à évaluer.

Finalement, on peut se demander si traiter le problème d'une manière différente ne permettrait pas d'apporter des solutions plus adaptées au concepteur. En effet ces besoins d'analyse de sensibilité, de propagation d'incertitudes servent à expliquer et à qualifier la nature des solutions obtenues. Mais une des raisons pour lesquelles ces méthodes n'expliquent pas vraiment la nature des solutions résident aussi dans leur manière de formuler le problème. En effet, nous pensons que traiter le problème de la conception avion comme une boîte noire n'est pas une bonne solution, car de cette manière on empêche une ana-

lyse des compromis disciplinaires. Pour cela, il est nécessaire aujourd'hui de tendre vers des approches interactives qui aident les concepteurs à penser les problèmes différemment en considérant davantage les aspects disciplinaires, et en ne se concentrant pas uniquement sur des fonctions objectifs, pré-établies. Ainsi la décomposition du problème en sous problèmes a été envisagée par plusieurs approches, et principalement par la MDO (*Multi-Disciplinary Optimisation*. Pour ces approches, il s'agit, lorsqu'un problème multi-objectif est composé de plusieurs sous-fonctions (*disciplines*), d'essayer d'utiliser la structure du problème pour le résoudre.

2.2 L'optimisation multidisciplinaire (*MDO*)

Les premières applications en MDO ont proposé des méthodes travaillant sur l'intégration des valeurs de paramètres communs à plusieurs disciplines [3]. Par la suite et avec l'amélioration des moyens de calcul, ces méthodes se sont mises à considérer d'autres aspects comme le couplage d'outils d'optimisation, l'intégration des informations par bases de données, etc. Cependant, dans la plupart des cas, un processus global centralisant une partie de l'information demeure. Ainsi au cours des dernières années, la MDO a donné lieu à plusieurs axes de recherche :

- l'analyse de l'impact de la décomposition du problème général en sous-problèmes ;
- le développement d'outils d'analyse (création de modèles d'approximation, robustesse de la simulation, l'analyse de sensibilité du résultat, aide à la décomposition du problème) ;
- l'étude des processus d'optimisation à utiliser en local.

Des solutions sont aujourd'hui industrialisées ou en cours d'industrialisation [21][51]. Elles adressent principalement les problèmes d'interopérabilité entre les outils de simulation et mettent à la disposition des utilisateurs une assistance à la décomposition de leur problème, un ensemble d'outils d'optimisation locaux performants ainsi que des moyens d'analyse de sensibilité du résultat.
La recherche s'oriente donc actuellement davantage vers l'étude des algorithmes de couplage d'information, vers la décomposition automatique du problème, ainsi que sur la qualité et la robustesse de l'exploration de l'espace.

2.2.1 Problématique du point fixe

Comme introduit précédemment, les premiers algorithmes ont consisté essentiellement à coupler des systèmes, à les exécuter et à optimiser les résultats. Dans la plupart des cas, l'exécution du couplage nécessite plusieurs itérations du processus à cause des interdépen-

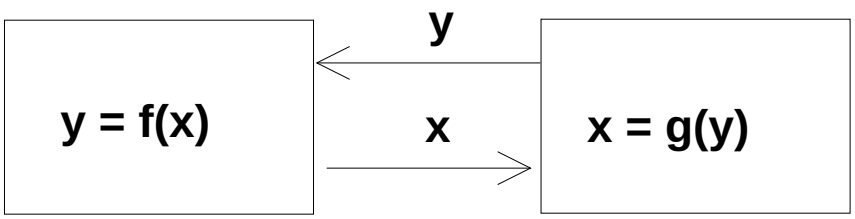


FIG. 2.5 – Couplage de deux systèmes

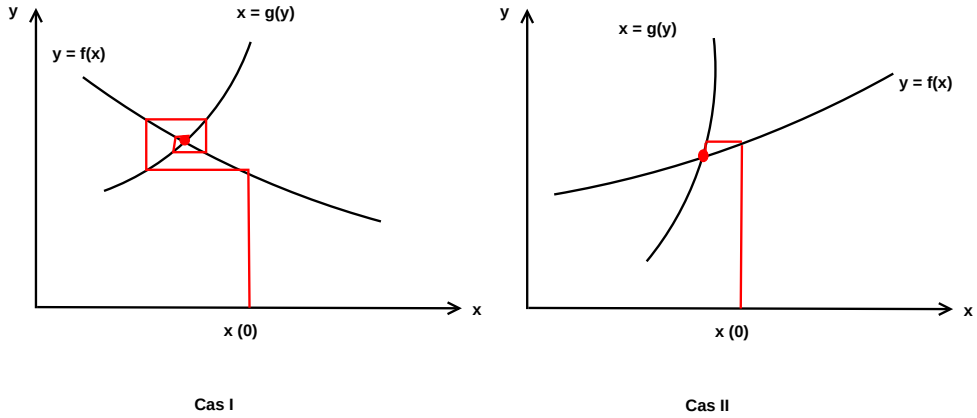


FIG. 2.6 – Itération du couplage de deux systèmes et convergence

dances entre les disciplines qui génèrent des boucles de rétroactions (cycles). Les valeurs obtenues en sortie ne sont alors cohérentes que lorsque les valeurs impliquées dans des boucles ont convergé. Les algorithmes du point fixe (FPI, *Fixed Point Iteration*) itèrent une série d'exécutions et ce jusqu'à trouver un ensemble de points de convergence aux boucles de rétroaction. La figure 2.5 présente le couplage de deux systèmes simples mais fortement dépendant l'un de l'autre. Le problème de ce type d'algorithmes est de s'assurer que le système converge vers un état, pour lequel les valeurs de rétroaction sont stables. En prenant l'exemple des figures 2.5, 2.6 et 2.7, l'objectif est de trouver des valeurs stables pour les paramètres x et y[1]. La figure 2.6 illustre deux situations de convergence, pour lesquelles on a choisi un état initial $x(0)$ puis itéré les deux fonctions jusqu'à obtenir des valeurs stables pour x et y. À l'inverse, la figure 2.7 expose un cas de divergence et un cas de convergence vers un minimum local, lorsque l'objectif du problème est d'obtenir les valeurs minimales pour x et y. Ces résultats montrent que ce genre d'approche est dépendant de la forme des fonctions ainsi que des conditions initiales choisies.

Ces exemples très simples sur deux systèmes illustrent aussi la difficulté de l'optimisation multidisciplinaire, puisque dans un système complet, les couplages n'apparaissent pas

[1]On considère que le système trouve des valeurs stables lorsque les valeurs obtenues successivement pour x et y sont identiques ou très proches.

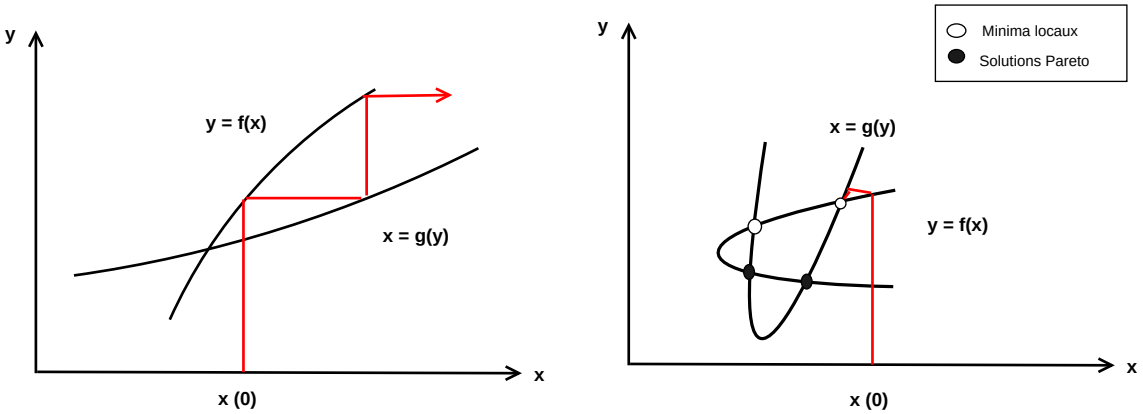


FIG. 2.7 – Divergence et minima locaux

seulement de système à système mais aussi à l'intérieur de boucles composées de chaînes d'exécutions. Dans ce cas, les comportements sont souvent complexes et non linéaires.

2.2.2 Les algorithmes à un niveau

Multi-Disciplinary Feasible (MDF)

L'approche MDF utilise directement l'algorithme FPI *Fixed Point Iteration*. Elle est composée d'un solveur (analyseur) et d'un optimiseur. Le solveur assure la cohérence entre les disciplines impliquées dans la simulation et propose des valeurs aux paramètres objectifs en fonction des paramètres de conception souhaités. Pour cela, il réalise une résolution de systèmes, qui garantit l'égalité sur les paramètres impliqués dans une boucle de rétroactions. Par exemple dans le cas de la conception avion, il assure la cohérence du bouclage *Masse/Performance*. Une fois cette analyse effectuée le solveur informe l'optimiseur des valeurs de performances obtenues. L'optimiseur se charge alors de modifier les valeurs des paramètres de conception afin d'améliorer les performances de l'avion.

Dans ce cas, l'optimisation est donc indépendante de l'exécution, et malheureusement le système MDF hérite de l'ensemble de problèmes issus de l'algorithme FPI. En particulier, les résultats obtenus ne sont pas toujours optimaux, car le solveur de système peut converger vers des minima locaux ou diverger, ou encore n'avoir aucune solution à proposer pour le paramétrage fournit par l'optimiseur. Mais au-delà de ces problèmes, l'optimisation étant complètement découplée du système, chaque modification de l'optimiseur implique une nouvelle résolution complète et donc des ré-exécutions et des réitérations qui peuvent être inutiles.

Pour résoudre les problèmes liés à la convergence vers des minima locaux et aux phénomènes de divergences, des alternatives ont été proposées. Il s'agit par exemple de changer l'ordre de la séquence utilisée par le solveur à chaque nouvelle résolution. Cependant, la

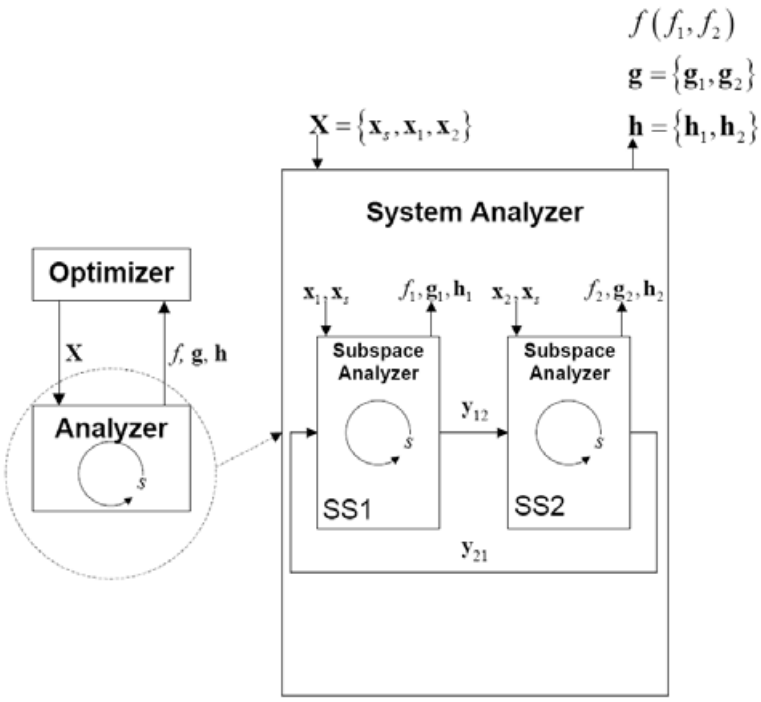


FIG. 2.8 – L'approche MDF

plupart des limites liées à FPI et au temps de calcul demeurent et l'utilisation de ce type de système reste limitée à des problèmes, dont les couplages et les temps de calculs sont faibles [46].

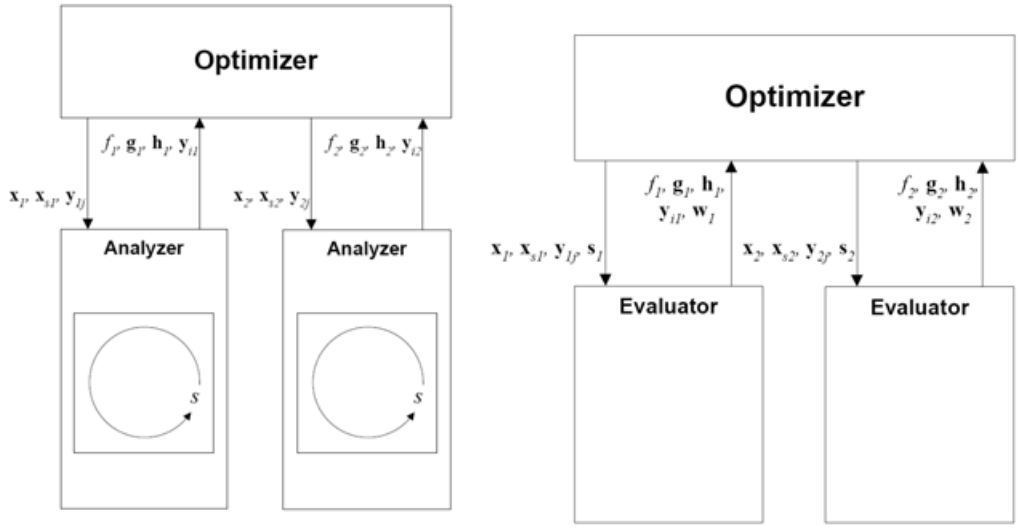
Individual Disciplinary Feasible (IDF)

IDF permet de décomposer la complexité du système en éclatant la séquence de calcul en sous-séquences, tel qu'illustré en figure 2.9(a). En comparaison avec MDF, chaque sous-système possède un solveur et l'optimiseur est maintenant aussi responsable de la coordination des sous-systèmes. IDF a amélioré MDF :

- en robustesse, car l'espace des solutions est mieux parcouru par chaque discipline ;
- en vitesse de convergence, l'optimiseur choisit des valeurs de variables partagées plus judicieuses ;
- en temps de calcul, les sous-systèmes sont exécutés en parallèle et peuvent donc être très facilement distribués.

Le système converge lorsque les variables partagées obtiennent des valeurs très similaires. Un inconvénient de ce type de système est que le produit en cours de conception n'a pas de cohérence (réalité physique) tant que les variables partagées n'ont pas convergé ; alors que MDF a une réalité physique puisque l'égalité de valeurs des variables partagées est garantie à chaque itération. La formulation mathématique d'IDF est proche de celle

de MDF, mais de nouvelles contraintes sont ajoutées afin d'assurer la convergence des variables partagées par plusieurs sous-systèmes.



(a) IDF (b) AAO

FIG. 2.9 – Les approches AAO et IDF

All At Once (AAO)

Cette méthode centralise l'analyse et l'optimisation. Alors que dans les méthodes précédentes on utilisait systématiquement un (MDF) ou plusieurs (IDF) solveurs de système, dans AAO on considère qu'un optimiseur peut simultanément optimiser les performances et résoudre les dépendances physiques (tel que le bouclage *Masse/Performance*). Ce type de résolution est donc très centralisé et l'optimiseur fait uniquement appel à des évaluateurs de systèmes capables d'exécuter des fonctions. L'optimiseur possède un vecteur de variables supplémentaires pour raisonner, qui l'informe sur l'état de chacun des évaluateurs. La différence principale est que cette fois-ci le système se passe complètement d'une résolution mathématique de système pour converger [46]. Ainsi les égalités sur les paramètres partagés, ainsi que les contraintes liées aux interdépendances physiques ne sont réalisées qu'à la convergence du système.
Finalement, appliquer une approche AAO revient à traiter le problème comme un problème d'optimisation multi-objectif, tel que précédemment.

	AAO	**IDF**	**MDF**
Utilisation d'analyseurs de système	Non	Oui	Oui
Cohérence des résultats	à la convergence	à la convergence	à chaque itération
Cohérence des sous-systèmes	à la convergence	à chaque itération	à chaque itération
Variables gérées par l'optimisation	conceptions, couplages des systèmes, états des sous-systèmes	conceptions, couplages des systèmes	conceptions

TAB. 2.1 – MDF IDF et AAO : Trois approches d'optimisation multidisciplinaires

2.2.3 Les algorithmes multi-niveaux

Le tableau 2.1 résume les 3 techniques présentées. Afin de les améliorer, plusieurs stratégies intégrant différents niveaux ont été proposées :

- *Concurrent Subspace Optimization* (CSSO)
- *Bi-Level Integrated System Synthesis* (BLISS)
- *Collaborative Optimization* (CO)

L'algorithme *"Collaborative Optimization"* (CO)

Une des limites des approches précédentes est leur optimisation centralisée qui est parfois peu adaptée à la réalité, qui est souvent décentralisée. L'optimisation collaborative propose donc d'utiliser une organisation à deux niveaux et de permettre au niveau inférieur d'utiliser un algorithme d'optimisation adapté à chaque sous-problème traité. Ainsi en local, chaque optimiseur est responsable d'un problème d'optimisation souvent multi-objectif mais non couplé. Cette décomposition redonne de l'autonomie aux sous-systèmes, et permet de diminuer les échanges d'information au niveau supérieur. L'optimiseur global est lui chargé de respecter ses critères d'optimisation et de trancher sur la valeur des variables partagées par plusieurs domaines. Il cherche donc à mener les sous-systèmes dans un espace de consensus dans lequel ses objectifs sont respectés. Un optimiseur de sous-système cherche alors à réaliser ses objectifs tout en satisfaisant ses contraintes. Il ne communique jamais avec ses pairs, mais s'en remet directement à l'optimiseur principal. Les contraintes sont donc décrites de la manière suivante :

- des *contraintes de convergence* sur les paramètres partagés par plusieurs domaines sont décrites au niveau du système ;
- des *contraintes sur les entrées partagées* et non partagées sont écrites dans les sous-

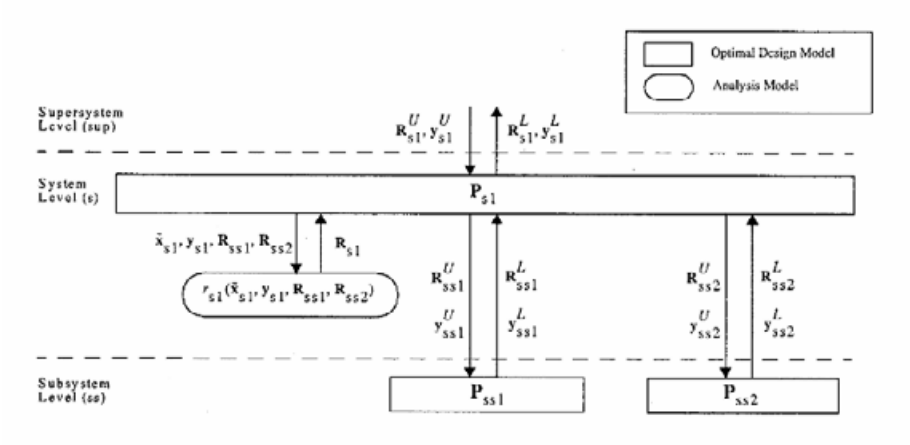


FIG. 2.10 – L'approche *Target Cascading*

systèmes ; mais les contraintes sur les entrées partagées peuvent changer au fil des exécutions selon les mises à jour du niveau système ;
– des *contraintes sur les valeurs de sorties* sont connues au niveau des sous-systèmes et parfois au niveau du système global.

L'algorithme par *Target Cascading*

Cette stratégie provient plutôt du monde automobile [49]. Elle s'inspire de la décomposition du produit en composants principaux. Elle est entièrement hiérarchique, mais contrairement aux approches précédentes, elle ne nécessite pas la mise en place d'un processus centralisé. Chaque composant possède ses objectifs, ses relations avec les autres et cherchent à minimiser une fonction représentant les objectifs demandés par le niveau supérieur ainsi que les contraintes remontées par les niveaux inférieurs. Néanmoins, de par sa structure, ce type d'algorithme est difficilement utilisable dans un contexte aéronautique, puisque les interdépendances entre les métiers et les composants y sont beaucoup plus fortes, ce qui implique qu'une décomposition purement hiérarchisée est difficilement réalisable, notamment à cause du bouclage masse/performance.

2.2.4 Synthèse sur MDO

La MDO propose des approches intéressantes [12] pour la conception avion, car elle tient compte de problématiques métiers. En cherchant à décomposer et à intégrer des sous-problèmes d'optimisation, elle considère une partie des relations disciplinaires et surtout utilise la structure du problème pour trouver des solutions, ce que ne fait pas l'optimisation multi-objectif (MOO).
Cependant cette considération disciplinaire se fait essentiellement à travers une décom-

position hiérarchique et mathématique du problème, qui influence nécessairement la résolution [2] et limite la compréhension globale du problème. En particulier, l'autonomie accordée aux disciplines n'est que relative, puisque la résolution se fait essentiellement par l'intégration des résultats à un niveau supérieur. Les résultats ne permettent donc pas au concepteur de découvrir l'interdépendance des contraintes au-delà du découpage préétabli, ce qui cache une partie des compromis disciplinaires et des relations entre les paramètres.
D'autres méthodes d'optimisation intégrant des connaissances distribuées ont été étudiées. Ces méthodes issues de l'Intelligence Artificielle (IA) sont étudiées dans le but de fournir des algorithmes génériques permettant la résolution de problèmes distribuées, c'est ce que nous allons voir dans la section suivante.

2.3 L'optimisation combinatoire à base d'agents

L'utilisation grandissante d'Internet, l'intégration de ressources logicielles distribuées ont fortement participé à l'étude de problèmes, pour lesquels les données, les contraintes et les processus de décision sont distribués. Des méthodes de résolution de problèmes distribuées ont donc été souhaitées [31] entre autre pour éviter des coûts dus à la centralisation des connaissances et des raisonnements, à des fins de confidentialité des informations, ainsi que dans un objectif de robustesse.
Prenons l'exemple de la planification d'une réunion, où chaque participant possède un agenda personnel et des préférences complexes, il n'est pas toujours possible pour des raisons de confidentialité, d'hétérogénéité des données, de centraliser l'ensemble des informations pour résoudre ce problème. D'autre part l'ajout d'une nouvelle personne ou de nouvelles contraintes, la défaillance d'un agenda doivent pouvoir être gérés sans avoir à redéfinir une formulation complète du problème qui tienne compte de ces modifications.
Des approches composées d'entités autonomes appelées agents, ont été étudiées. Leur principe est que chaque entité contient une partie du problème et collabore avec ses pairs pour parvenir à une solution globale. Ainsi des algorithmes de satisfaction de contraintes dont le raisonnement et les connaissances sont distribués, ont été proposés.

2.3.1 La notion d'agent

Un agent est une "entité virtuelle ou réelle qui est capable d'agir sur son environnement, qui possède des moyens de perception et de représentation partielle de son environnement, qui est capable de communiquer avec d'autres agents et qui est autonome dans sa prise de décision" [32]. De façon générale, un agent possède donc principalement :

- des *compétences* : ce qu'il sait faire ;
- des *croyances* : ses connaissances sur son environnement, sur les autres, et parfois sur lui-même ;
- des *accointances* : relations sociales, l'ensemble des agents qu'il côtoie ;
- des *aptitudes* : capacités de perception, d'action, de décision.

Plongé dans un environnement, l'agent doit utiliser l'ensemble de ces caractéristiques pour répondre aux attentes du concepteur.

2.3.2 La formulation des problèmes de DisCSP

La satisfaction de contrainte (CSP) est un problème classique en IA depuis de nombreuses années : on le retrouve dans diverses applications logicielles ou industrielles telles que l'affectation de configuration et la planification de ressources. De ce fait, c'est majoritairement sous cet angle que les algorithmes d'optimisation ou de satisfaction de contraintes distribuées ont été étudiés (Distributed Contraint Satisfaction Programming : DisCSP) [77].
Un problème de CSP est composé de n variables $\{x_1, x_2, ...x_n\}$, aux valeurs discrètes et finies $\{D_1, D_2, ...D_n\}$ et d'un ensemble de contraintes sur ces valeurs. Cet ensemble peut être décrit sous la forme de *nogoods*, définissant des incompatibilités de valeurs entre les x_i. Une solution au CSP est alors un ensemble de valeurs ne comportant aucun *nogood*. Les problèmes de CSP sont connus pour être NP-complet. Dès lors que les variables et les contraintes sont distribuées parmi des entités/agents, on parle alors de DisCSP. Ils sont composés comme suit :

- un ensemble d'agents, $1, 2, ..., k$
- à chaque variable de CSP est associée un agent.
  - un ensemble de valeurs associées à chaque variable ;
  - un ensemble de contraintes intra-agent, propres à la valeur de la variable ;
  - un ensemble de contraintes inter-agent, relatives aux incompatibilités de valeurs.

Ainsi une solution au DisCSP est une situation dans laquelle les agents ont trouvé des jeux de valeurs pour lesquels aucune contrainte intra/inter-agent ne soit violée.

2.3.3 Les algorithmes de DisCSP basés sur des *nogoods* : ABT et AWC

De par leur formulation, les algorithmes de résolution de DisCSP ont été fortement influencés par les algorithmes de CSP. Pour des questions de performance, la plupart de ces algorithmes sont asynchrones. Cette caractéristique permet en effet à un agent de ne pas attendre un message particulier avant de s'exécuter ou d'agir. D'une manière plus

générale, on peut dire que les algorithmes de DisCSP doivent répondre à deux critères : produire des applications performantes en temps, et en parcours d'espace.

Asynchronous BackTracking (ABT)

L'algorithme ABT [4] utilise la formulation du DisCSP introduite précédemment. Le réseau d'agents est hiérarchisé (ordre total), de telle sorte qu'un agent prédécesseur envoie la valeur qu'il choisit à ses successeurs. Les agents successeurs utilisent alors cette valeur et renvoient un message *ok*? si la valeur reçue est acceptable avec les contraintes de l'agent ou un message *nogood* indiquant à l'agent prédécesseur sa non satisfaction de la situation. Les messages *nogood* contiennent également des informations relatives au contexte de l'échec, qui permettent aux agents de mémoriser les actions qui ont échoué, et de progressivement tendre vers une satisfaction des contraintes.

Asynchronous WeakCommitment (AWC)

Dans AWC [76], les agents utilisent les mêmes principes que pour ABT, mais la transmission des *nogoods* y est légèrement différente, dans la mesure où les agents utilisent des niveaux de priorité. Avant de générer des *nogoods* et dans un premier temps, chaque agent choisit toujours une valeur pour sa variable, qui soit conforme avec les agents voisins possédant un plus haut niveau de priorité, ce qui réduit le nombre de conflits aux voisins prioritairement inférieurs. Dans le cas où un agent ne peut pas trouver une valeur cohérente avec son environnement, il élève sa priorité d'une unité au-dessus de la priorité maximale de ses voisins et génère un *nogood* qu'il envoie à son voisinage. Ainsi les situations les plus conflictuelles sont progressivement traitées.

ABT et AWC sont donc focalisés sur la manière de stocker et d'utiliser les *nogoods*. Au cours de la résolution les agents mémorisent ces *nogoods* et réutilisent leurs expériences passées pour converger vers une solution. De ce fait ABT et AWC sont garantis complets. Mais, ils nécessitent *a priori* un nombre important de *nogoods*, qui sont par conséquent coûteux en mémoire. Ce problème peut être très contraignant, dès que les cas à traiter deviennent conséquents et que la quantité mémoire est limitée [44].

2.3.4 Les *Distributed Breakout Algorithms* (DBA)

D'autres algorithmes ont alors été proposés afin de s'affranchir de cette utilisation de *nogoods*. C'est le cas des *Distributed Breakout Algorithms* [44]. Ils consistent simplement à donner des poids aux différentes contraintes et à essayer en permanence de diminuer la somme de ces poids. Ces algorithmes sont donc non complets et il n'est pas possible de

Agent 1	**Agent 2**
Variables x_1, x_2	Variables x_3, x_4
$C_1 : x_1 \vee x_2$	$C_3 : x_3 \vee x_4$
$C_2 : \neg x_1 \vee \neg x_2$	$C_4 : \neg x_3 \vee \neg x_4$
$C_5 : \neg x_1 \vee \neg x_3$	$C_5 : \neg x_1 \vee \neg x_3$
$C_6 : \neg x_2 \vee \neg x_4$	$C_6 : \neg x_2 \vee \neg x_4$

TAB. 2.2 – Un problème de DisSAT

garantir la globalité du minima trouvé. Dans certains cas le système converge vers des minima locaux qui sont définis de la façon suivante :

– un état est un minimum local, s'il existe une solution satisfaisante au problème alors qu'au moins un des agents a une de ses contraintes non satisfaite et qu'aucun changement local ne semble permettre l'amélioration de cet état de non satisfaction.

Des techniques pour sortir de ces minima locaux ont été proposées, elles consistent à modifier les poids des contraintes non satisfaites, afin de modifier certains équilibres et permettre ainsi aux agents de trouver d'autres espaces admissibles.

La proposition Multi-Distributed Breakout (DB)

Les approches DisCSP sont coûteuses en mémoire et en parcours d'espace de recherche, elles sont donc difficilement applicables à des problèmes réels. Les gains en terme de complexité des DBA ont permis d'aborder d'autres problèmes types, pour lesquels un agent n'a plus seulement la responsabilité d'une unique variable mais de plusieurs. D'autre part, des contraintes locales entre ces différentes variables peuvent être attribuées à chaque agent, ce qui lui confère plus d'autonomie et permet donc d'adresser des problèmes plus réalistes, et surtout plus proches d'une approche agent, où chaque individu possède des contraintes et un mode de raisonnement. Le tableau 2.2 présente un de ces problèmes types à 4 paramètres (x_1, x_2, x_3, x_4) pour lequel chaque agent possède des clauses locales (C_1,..., C_6) et des degrés de libertés : x_1, x_2 pour l'agent 1 et x_3, x_4 pour l'agent 2.

L'algorithme Multi-DB [43] possède les caractéristiques suivantes :

– Premièrement, chaque agent utilise une résolution locale pour choisir la valeur qu'il compte modifier. Étant donné que l'agent est maintenant multi-valeurs, il possède un ensemble de contraintes à satisfaire, cet ensemble de contraintes impactant lui-même d'autres contraintes. Il doit donc choisir une action qui lui permet de minimiser la somme des contraintes, qu'il perçoit. Pour cela, Multi-DB utilise des algorithmes locaux d'optimisation tels que la recherche tabou, l'apprentissage, etc. Chaque agent a la possibilité d'utiliser cet algorithme et de faire les choix qui lui semblent judicieux.
– Le fait d'utiliser une approche non complète ne permettant pas d'identifier le contexte

de chaque modification, rend difficile une approche multi-variable. Il est en effet souvent nécessaire d'utiliser un mécanisme d'exclusion mutuelle des modifications pour évaluer l'impact d'une action à la fois. En cas d'échec d'une modification, il est alors plus facile d'identifier la provenance de l'augmentation des contraintes non satisfaites. Il est alors aussi plus facile d'identifier l'agent responsable puisqu'une seule modification a été effectuée. Dans Multi-DB, l'exclusion mutuelle n'est pas nécessaire, mais un agent observateur est capable d'intervenir en cas d'augmentation de contraintes non-satisfaites pour favoriser l'identification de l'agent qui en serait responsable.

– Enfin, dans Multi-DB, chaque agent est capable de se réinitialiser pour recommencer une résolution en cas de non convergence et d'insatisfaction de ses contraintes.

Ce type d'algorithme est intéressant, car il permet de converger plus rapidement que les algorithmes complets et offre plus de souplesse dans la formulation du problème. Des améliorations ont été proposées, et leurs performances comparées en terme de nombre de cycle de résolution, somme des modifications, etc. Cependant, la non complétude ne permet pas de s'assurer que le système va trouver la solution, et l'approche ne permet pas non plus de chercher à pondérer des contraintes en introduisant des fonctions de coût. C'est pourquoi tous les algorithmes que nous avons présentés jusqu'à maintenant adressent des problèmes de satisfaction de contraintes et ne cherchent jamais à optimiser les solutions. Ils ne permettent pas, par exemple, de définir des fonctions de coût liées à des contraintes partagées par deux agents.

2.3.5 De la satisfaction de contraintes à l'optimisation

L'introduction d'une fonction de coût

Pour prendre en compte des fonctions de coût, une nouvelle formulation du problème DCOP (*Distributed Constraints Optimization Problem*) a été proposée [11] ; elle reprend les éléments de bases de DisCSP. Un DCOP est composé de n variables, $\{x_1, x_2, ..., x_n\}$, associées à un ensemble d'agents. Les variables x_i peuvent prendre n'importe quelle valeur de leur domaine D_i. L'objectif est de choisir des valeurs de variables qui minimisent une somme de coût. Ainsi un coût est associé à chaque couple de valeurs voisines $f(d_i, d_j)$ et la somme des $f(d_i, d_j)$ est à minimiser.

Dans l'exemple de la figure 2.11, $\{x_1, x_2, x_3, x_4\}$ sont des variables avec pour domaine $D = \{0, 1\}$. Les fonctions de coût associées sont décrites dans le tableau. D'autres contraintes que nous ne détaillerons pas ici permettent de définir des fonctions de coût pour des contraintes $n-aires$. Par exemple, la fonction g associe des contraintes ternaires à des valeurs de paramètres.

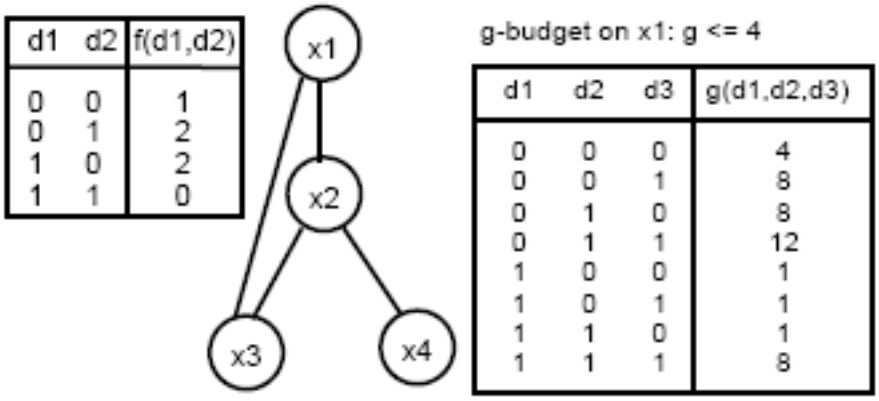


FIG. 2.11 – Graphe de DCOP

ADOPT : un algorithme asynchrone d'optimisation distribuée

Utilisant cette formulation DCOP, l'algorithme ADOPT *Asynchronous Distributed Constraint Optimization* a été proposé. Il est complet, asynchrone et a été largement discuté dans la littérature [58] [71]. Nous ne fournissons ici qu'une courte description.

Comme les algorithmes ABT et AWC, ADOPT commence par hiérarchiser la structure du problème, en l'organisant sous la forme d'un arbre de *Depth-First Search* (DFS), dans lequel uniquement les contraintes entre variables précédentes et descendantes sont possibles. En particulier, des contraintes avec des variables appartenant à d'autres sous-arbres ne sont pas possibles. Il est donc difficile de traiter des problèmes pour lesquelles les variables ont des contraintes $n - aires$. La figure 2.12 illustre cette transformation d'un réseau de contraintes en un arbre DFS. On remarque que dans l'arbre DFS, x_2 est considéré comme un enfant de x_1, alors que x_3 est un descendant (mais pas un enfant) de x_1. Cette différence est importante puisque lors de la résolution x_3 reçoit les valeurs de x_1 mais ne lui adressera aucune demande directe. Finalement, ADOPT est organisé selon les principes suivants :

- À l'initialisation, les valeurs sont choisies aléatoirement pour chaque noeud.
- Des messages de valeurs sont envoyés des prédécesseurs vers leurs enfants et éventuellement vers certains descendants (tel que x_1).
- Des messages de coût sont envoyés des enfants vers les parents indiquant le coût du sous-arbre de l'enfant (par exemple, x_3 enverra son coût à x_2 et x_2 à x_1).
- Enfin des messages contenant des seuils minimal et maximal sont également transmis entre les noeuds père et enfant. Ces seuils permettent à un père de mesurer les écarts entre ses sous-arbres. Ainsi lorsque les seuils minimal et maximal sont égaux, le système a convergé.

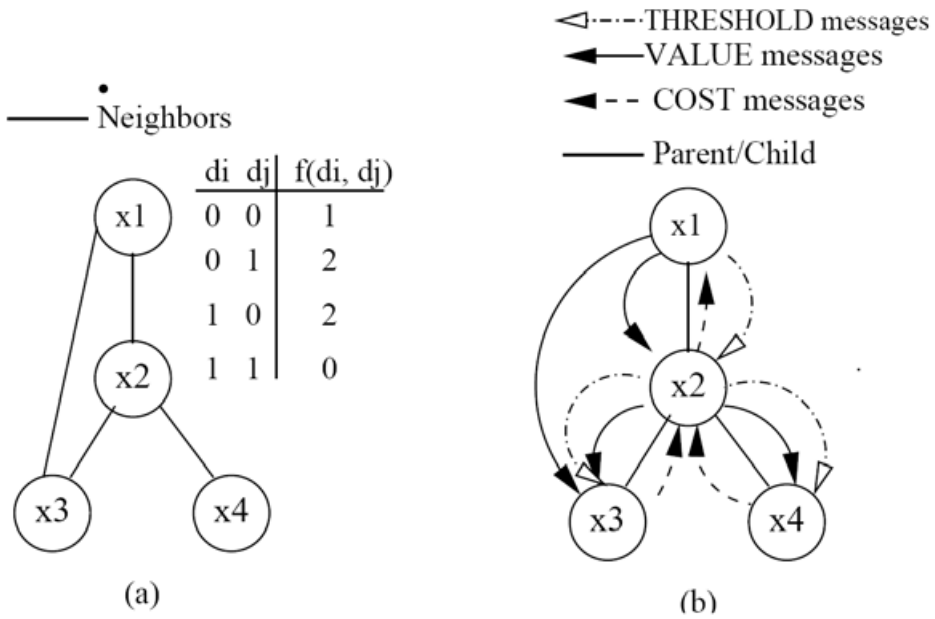


FIG. 2.12 – Transformation d'un problème de DCOP en un arbre DFS

La complétude d'ADOPT est prouvée [58]. Néanmoins comme ABT et AWC, il utilise une hiérarchisation du problème, adresse des problèmes discrets et ne permet pas de définir simplement des contraintes liant plus de deux variables. Il présente donc des limites en termes de passage à l'échelle et de généricité.

2.3.6 Synthèse des approches combinatoires à base d'agents

Les approches de satisfaction de contraintes ou d'optimisation sont étudiées depuis une dizaine d'années et offrent des résultats intéressants en termes de distribution, d'asynchronisme, de performances, etc. Cependant ces approches restent majoritairement académiques et peu d'applications réelles ont été adressées. Si elles permettent de résoudre des problèmes d'ordonnancement et de coordination, elles présentent de nombreuses limites pour la résolution de problèmes distribués et ce pour plusieurs raisons :

- les variables utilisées sont discrètes ;
- la définition des contraintes est explicite et locale, et une fonction de coût est à donner pour chaque couple de valeurs ;
- les contraintes duales sont préférables, et le passage à des contraintes n-aires est encore à l'étude ;
- pour assurer leur complétude, ces algorithmes sont basés sur des méthodes statiques, qui ne permettent pas la modification du problème au cours de la résolution.

Or dans les problèmes réels :

- de nombreuses contraintes sont souvent implicites et dues aux objectifs locaux et aux connaissances propres à chaque agent. Ainsi si les agents ont des préférences locales explicites sur leurs objectifs, il n'est pas souvent possible de définir des fonctions de coût associées à des couples de variables. Le seul moyen de connaître ces contraintes implicites est donc souvent d'utiliser la dynamique du système ;
- la formulation des problèmes réels intègrent la plupart du temps des paramètres continus ;
- en conception de produit complexe, l'objectif n'est pas forcément de rechercher une solution optimale à un problème statique et complètement défini. Souvent le concepteur veut changer ses contraintes et son système de manière dynamique. Les algorithmes complets sont donc mal adaptés aux problèmes dynamiques pour lesquels des agents peuvent être ajoutés/modifiés/supprimés.

Néanmoins ces algorithmes posent les bases de la résolution de problèmes distribués à base d'agents, comme le besoin dans ces approches d'utiliser des procédures de *backtracking* [64] ainsi qu'une mémoire des états rencontrés précédemment. De plus, les algorithmes *distributed breakout algorithms* ont permis d'aborder d'autres problèmes types, pour lesquels un agent n'a plus seulement la responsabilité d'une unique variable mais de plusieurs, et où ils commencent donc à avoir un raisonnement plus complexe en émettant des préférences.

2.4 Synthèse de l'existant en conception avion

Les méthodes traditionnellement utilisées pour résoudre la conception préliminaire avion sont basées sur des méthodes d'optimisation globales ou hiérarchiques.

En tant que méthode globale multi-objectif (MOO), les algorithmes génétiques ont montré leur efficacité à traiter ce problème, parce qu'ils sont robustes aux discontinuités et assez indépendants des caractéristiques de la fonction globale à optimiser. Le principal inconvénient de cette méthode d'optimisation globale est qu'elle ne fournit qu'une vue limitée sur les relations d'interdépendances liant l'ensemble des paramètres [6] et aucune sur les compromis disciplinaires. Aussi, dans le cas où le problème est sur-contraint le concepteur n'a aucune aide ni pour reformuler son problème, ni pour agir sur le système. De plus, la plupart de ces approches étant stochastiques, elles deviennent plus difficilement utilisables, lorsque le nombre d'objectifs et/ou de degrés de libertés augmentent.

Les méthodes d'optimisation multi-disciplinaires (MDO) [12] sont des approches intéressantes, car elles cherchent à décomposer et à intégrer des sous-problèmes d'optimisation. Pour cela, elles considèrent les disciplines en tant qu'éléments agissant les uns avec les autres. Cependant elles résolvent le problème en utilisant une décomposition et une

intégration hiérarchique des sous-problèmes, ce qui influence les résultats [2] et limite la compréhension globale du problème. En particulier, les résultats n'aident pas le concepteur à découvrir l'interdépendance des contraintes au-delà du découpage préétabli, ce qui cache une partie des compromis disciplinaires et des relations entre les paramètres.

Les approches d'optimisation (DCOP) et de satisfaction de contraintes distribuées (DisCSP) [42] [58] [53], ont montré l'intérêt de l'optimisation distribuée à l'aide d'agents, et ont donné lieu à de nombreux travaux, comparant méthodes complètes/non complètes et illustrant l'importance d'une phase de retours arrières et de la mémoire. Mais la formulation particulière de ces approches les rend difficilement transposables à des problèmes continus et non-linéaires.

Concevoir un avion, ne consiste pas uniquement à résoudre un problème d'optimisation, mais plutôt à faire une succession de compromis inter-disciplinaires qui permettent de préciser incrémentalement les caractéristiques du produit [6]. Le concepteur est donc souvent amené à modifier ses objectifs afin de converger vers une solution. La conception préliminaire est donc un processus dynamique et évolutif qui nécessite souvent des ajustements sur les objectifs à atteindre, des changements de granularité des fonctions mathématiques utilisées, etc. Pour accompagner le concepteur dans sa tâche, nous proposons donc d'aller vers des approches dynamiques et interactives, et de nous détacher d'un contexte d'optimisation pour aller vers des approches qui soient capables d'intégrer plus de connaissances métiers, qui apprennent des relations entre disciplines, qui permettent d'enrichir les connaissances sur les modèles utilisés.

Mais pour intégrer et utiliser ces caractéristiques, la fonction de simulation avion doit être considérée comme un système complexe, composée de non-linéarités, d'éléments disciplinaires et d'interdépendances fortes entre les objectifs de ces éléments. Un moyen d'aborder ce problème est donc de le modéliser comme étant un système aux propriétés complexes et d'utiliser son comportement pour apprendre les interdépendances et pour trouver des compromis inter-disciplinaires satisfaisant l'ensemble des contraintes. Les intérêts à considérer le problème de cette manière sont les suivants :

- on utilise sa structure pour le résoudre et donc ainsi la résolution devrait être fortement discriminante par rapport aux approches globales de type MOO ;
- on favorise la recherche de consensus et la prise en compte des connaissances et exigences propre à chaque discipline. Ainsi la résolution devrait apporter une meilleure compréhension du problème ;
- on s'autorise un maximum de liberté dans la formulation du problème, ce qui doit permettre la modification de contraintes au cours de la résolution.

Deux dangers ne cessent de menacer le monde ; l'ordre et le désordre.

Paul Valéry

3

Le contexte théorique et scientifique

Sommaire

Dans cette partie, nous introduisons quelques définitions sur la notion de simulation ainsi que sur les systèmes complexes. Après avoir présenté ces concepts de haut niveau et la manière dont la systémique et la cybernétique proposent d'aborder les systèmes complexes, nous montrons comment les systèmes multi-agents et plus particulièrement la théorie des AMAS offrent un cadre théorique pour en modéliser/simuler le fonctionnement.

3.1 Les exigences de la simulation des systèmes complexes

3.1.1 La simulation : introduction et définition

La simulation est devenue un terme tellement universel, qu'il est difficile d'en donner une définition générale.

- Shannon (1998) introduit la simulation comme : *"the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behavior of the system and/or evaluating various strategies for the operation of the system"*
- Pour Fishwick (1996) : *"Computer simulation is the discipline of designing a model of an actual or theoretical physical system, executing the model on a digital computer, and analysing the execution output"*

La définition de Shannon insiste sur la notion de système et de modèle. Le système correspond à l'ensemble des éléments qui peuvent interagir pour accomplir certains objectifs. Le modèle est une représentation sur la manière, dont les objets du système peuvent interagir. La simulation peut ainsi être vue comme une manière de mettre en fonctionnement les hypothèses et choix d'un concepteur faits sur un ensemble d'objets. Très similaire, la définition de Fishwick est néanmoins plus orientée vers l'introduction d'une méthode, composée de trois étapes principales : l'élaboration du modèle, son exécution et enfin l'analyse des résultats obtenus.

Finalement selon ces définitions, la simulation peut donc concerner tout système physique, social, naturel, réel ou virtuel, pour lequel l'observation et l'analyse de la dynamique de l'exécution permettent une compréhension des phénomènes. En général, l'utilisation de simulations se fait de manière inductive ou déductive.

- Dans le cas d'une utilisation déductive, on cherche à trouver une modélisation d'un phénomène réel. La démarche consiste à faire des hypothèses sur des éléments du système et à affiner ses hypothèses afin de reproduire le comportement naturel ou social observé.
- Dans la méthode inductive, la modélisation des éléments du système est *a priori* connue, on cherche alors à analyser ou prévoir le comportement engendré par la mise en relation des modèles. Cela peut concerner la recherche d'un paramétrage ou/et d'une organisation adéquate de ces éléments.

Mais en réalité les besoins d'une simulation se situent souvent entre ces deux modes, car la définition des éléments dépend de la dynamique de la simulation ainsi que la compréhension/définition du comportement global attendu. Par exemple, dans le cas de la

conception avion, la simulation est plutôt inductive, puisque il s'agit souvent de trouver un paramétrage des éléments qui permettent d'atteindre une série d'objectifs, néanmoins la réflexion sur le choix et l'affinage des modèles de simulation est basée sur le mode déductif. Or la dynamique de la résolution et l'évolution de la formulation du problème tel que le changement de choix technologiques ont nécessairement un impact sur les choix de modélisation.
Finalement, lorsque les éléments de la simulation sont fortement interdépendants et que le système simulé est complexe, alors la compréhension des résultats nécessite la mise en place d'approches qui puissent traiter les propriétés des systèmes complexes. Ces approches sont basées sur la définition des éléments de la simulation et sur la caractérisation des interactions entre ses éléments.

L'approche que nous proposons par la suite ne pourra pas être assimilé à une simulation de la conception préliminaire avion, puisque nous concentrerons nos efforts sur la résolution d'un problème. Cependant en considérant la fonction d'évaluation comme un système complexe et non comme une boite noire, on cherche à savoir ce qui s'y passe et on tend ainsi vers des approches de simulation.

3.1.2 Les systèmes complexes

Définition

J.-L. LeMoigne [52] donne d'un système complexe la définition suivante : *"un système complexe, c'est quelque chose qui poursuit des finalités dans un environnement actif et évolutif en exerçant une activité, en s'organisant et en évoluant sans perdre son identité".*

Il est composé d'éléments hétérogènes reliés par des interactions fortes mais fluctuantes. *"Des qualités émergentes naissent de l'organisation du tout et peuvent rétroagir sur les parties"*[60]. *"Un système complexe est par définition un système que l'on tient pour irréductible à un modèle fini, aussi compliqué, stochastique, sophistiqué que soit ce modèle, quels que soient sa taille, le nombre de ses composants, l'intensité de leurs interactions".*

Un système complexe est donc différent d'un système compliqué. Pour un système compliqué, la séquence de conception peut être difficile à maîtriser mais son séquencement est logique et itérable, alors que celle d'un système complexe ne l'est pas. Voici quelques facteurs qui corrélés augmentent la complexité des systèmes :

1. un nombre important de composants ;
2. des interdépendances fortes et nombreuses entre les composants ;
3. l'apparition de comportements non prévisibles ;
4. la présence de non-linéarité ;

Mais elle peut aussi être d'un autre niveau, tel que les mesures par l'hétérogénéité des connaissances au sens de Shannon-Wiener, ou par la complexité informationnelle (complexité par le nombre de paramètres, ou de symboles nécessaires pour caractériser le système).

L'étude des systèmes en systémique

La systémique est : *"une nouvelle discipline qui regroupe les démarches théoriques, pratiques et méthodologiques, relatives à l'étude de ce qui est reconnu comme trop complexe pour pouvoir être abordé de façon réductionniste, et qui pose des problèmes de frontières, de relations internes et externes, de structure, de lois ou de propriétés émergentes caractérisant le système comme tel, ou des problèmes de mode d'observation, de représentation, de modélisation ou de simulation d'une totalité complexe"*[26].

En systémique, l'intelligibilité du complexe se fait par la modélisation, et propose d'étudier les systèmes complexes en s'intéressant aux concepts et propriétés suivantes :

- **la globalité** : elle s'exprime dans le comportement du système, et se traduit par l'adage : *le tout est plus que la somme des parties.*
- **l'interaction** : ce concept essentiel complète bien celui de la globalité, car il permet d'étudier les relations entre les constituants du système à un niveau élémentaire. Cette étude fournit des informations sur les flux de matière, d'énergie, d'information qui peut être échangés de constituant à constituant.
- **la nature de l'information** : il existe en systémique deux types d'information, celles qui sont structurantes et celles qui sont circulantes. Une information structurante est incluse dans les mémoires du système, alors qu'une information circulante est un flux périssable à traiter. Dans certains cas, l'information circulante aide à construire ou à renforcer l'information structurante.
- **la rétroaction** : un système a des variables d'entrée et des variables de sortie. En général, les entrées sont sous l'influence de l'environnement du système et les sorties résultent de son activité interne. On appelle alors boucle de rétroaction (*feedback*) tout mécanisme permettant de renvoyer à l'entrée du système sous forme de données, des informations directement dépendantes de son activité et donc de ses sorties.

Provenant de l'automatique, la cybernétique a largement participé à l'émergence de ces concepts. Cette science se base sur l'étude des mécanismes à mettre en place pour permettre à des machines de s'autoréguler. Les notions de système, d'interactions entre les éléments d'un système, de *feedback*, d'autorégulation, de rétroaction, etc. sont donc tout aussi familiers à la cybernétique qu'à la systémique.

Les sciences de la complexité

Le rapprochement de la cybernétique, de la systémique et d'un nombre important d'autres disciplines (biologie [55], écologie et sociologie [59], informatique [48], philosophie [40] ou théorie des organisations [72]) ont abouti à l'émergence des sciences de la complexité. Comme la cybernétique et la systémique, le principe de la séparation fonctionnelle est maintenu (par disciplines, objets, sujets) et une attention particulière est portée aux relations entre ces séparations. Mais cette fois-ci, le schéma tient compte de l'ordre, du désordre et de l'organisation. En science de la complexité, on s'intéresse à toutes les propriétés et mécanismes qui permettent l'étude des systèmes complexes. En partant de la constatation que des phénomènes peuvent apparaître du désordre, les sciences de la complexité recherchent l'ordre par la stabilité, la régularité et la caractérisation du désordre (blocages, oscillations), en remettant en cause les choix organisationnels. Pour tenir compte de l'ordre, du désordre, l'organisation doit être flexible et doit permettre aux éléments du système de s'adapter. Cette notion a pris forme dans les années 80-90 au SFI[2] [48], elle vient de l'idée qu'une organisation qu'elle soit vivante, informatique, politique dépend de son environnement et se construit avec son environnement à travers des *feedbacks* positifs ou négatifs. C'est à partir d'eux que le système doit donc décider de ses déviations, qui sont souvent la cause des changements et parfois la conséquence d'évolutions irréversibles.

Dans *la méthode* [59], Morin caractérise et étudie ces systèmes. À travers ses propos,

1. il montre l'importance pour un système complexe d'avoir ses propres capacités d'auto-organisation et propose de regrouper l'ensemble des propriétés d'auto-* en trois catégories, celles qui permettent au système de s'auto-produire, celles qui assurent l'auto-régulation et enfin celles de l'auto-assemblage ;
2. il illustre également les limites de l'adage qui dit que le *tout est plus que la somme des parties*. En effet, un système n'est pas seulement plus que la somme des parties, dès lors que les propriétés d'une partie sont limitées par les contraintes de l'ensemble. Ainsi la notion d'organisation devient capitale, puisque elle permet simultanément l'apparition de propriétés globales, tout en contraignant beaucoup de propriétés locales.

Finalement en rompant avec la logique cartésienne qui tend à utiliser une approche analytique et réductionniste des problèmes en les décomposant simplement en plus petits éléments, ces approches ont apporté une nouvelle méthode *qui détecte et non pas occulte les liaisons, articulations, implications ou interdépendances* présentes dans les systèmes complexes [59]. Concrètement, toutes ces approches sont à la base de nombreuses idées et

[2]Santa Fe Institute

réflexions sur les caractéristiques des systèmes complexes et sur la manière de les aborder et modéliser, dont en voici quelques éléments.

Quelques éléments clés de ces approches

Relations locales et globales

Les relations locales à globales sont essentielles dans un système complexe, puisque l'on cherche à "penser global en agissant local". Ainsi les éléments locaux sont autonomes et prennent leur décision en fonction de leur perception locale. Selon les systèmes, les décisions sont plus ou moins réversibles. Dans le cas de simulation informatique, nous avons cet avantage fort mais aussi recherché de pouvoir prendre des décisions et de les ajuster en fonction du *feedback* environnemental.

L'organisation et les modèles de décision

Les études de Simon sur la rationalité humaine abordent le rôle de l'organisation sur la prise de décision. Il étudie en particulier la manière dont les individus peuvent agir ensemble dans l'organisation alors même qu'ils peuvent avoir des visions différentes du monde. La limitation de la rationalité individuelle illustre que *le milieu organisationnel et social dans lequel se trouve la personne qui prend une décision détermine les conséquences auxquelles elle s'attendra, celles auxquelles elle ne s'attendra pas ; les possibilités de choix qu'elle prendra en considération et celles qu'elle laissera de côté* [72].
Simon distingue deux types de rationalité : l'analyse de la rationalité substantive, qui considère qu'un individu est capable de prendre toutes les bonnes décisions en procédant uniquement à une analyse environnementale, et l'analyse de la rationalité procédurale qui considère que la perception environnementale est incomplète ou erronée sur la situation et qu'un acteur peut être incapable de calculer toutes les conséquences de ses actions.

La rationalité procédurale étudie donc le cheminement qui amène les individus à prendre des décisions, qui soient rationnelles. Cette analyse de Simon nous montre que finalement les deux types de rationalité sont complémentaires l'une de l'autre : la rationalité procédurale ayant un intérêt évident lorsque la rationalité substantive ne peut fournir de réponse.

Finalement, cette analyse renforce l'idée que le comportement d'un système dépend de la capacité de ses éléments à tenir compte de leurs connaissances individuelles, à apprendre de leur environnement et à s'adapter à leur perception environnementale, puisque l'organisation dans laquelle ils évoluent influence directement leur mode de décision.

La notion d'émergence

Concevoir des systèmes composés d'éléments distribués et autonomes et en simuler le comportement amène à définir la notion d'émergence [33].

L'émergence est un concept difficile à définir de par la variété des interprétations qu'on

en donne. Pour Bergson, l'émergence en tant que "phénomène" n'existe pas vraiment, et provient selon lui davantage d'un manque d'explications et de compréhension du "phénomène". Dans ce cas, on associe l'émergence à quelque chose qu'on ne comprend pas, et dont par conséquent on ne peut pas prédire le comportement. Limiter la notion d'émergence à des phénomènes que nous sommes incapables d'expliquer à l'heure actuelle semble donc délicat d'un point de vue scientifique, car non démontrable et probablement souvent lié aux limites de la connaissance scientifique. En suivant cette logique, des phénomènes perçus comme émergents à une époque sont finalement expliqués et compris par la suite.

Par conséquent, la définition de Goldstein est plus facilement acceptable et abordable ; c'est celle que nous utiliserons. Selon lui on peut parler d'émergence lorsque l'étude de la configuration et des interactions des composants d'un système offre plus d'explications sur les dynamiques et les phénomènes observés que les explications basées sur une analyse des parties seules. Les phénomènes observés sont dès lors caractérisés par le fait qu'ils ne sont ni prédictibles, ni réductibles aux seules parties.

Dans ce cas, l'émergence est donc un moyen descriptif des propriétés particulières d'un système macroscopique composé d'un ou de plusieurs niveaux microscopiques. Elle n'est donc plus tellement un phénomène inexplicable mais un phénomène particulier que l'on peut étudier en s'intéressant aux lois, aux interactions, aux connaissances qui régissent un niveau micro et qui permettent l'apparition de propriétés à un niveau macro. Finalement, on revient par cette définition à une analyse du comportement d'un système complexe, composé d'éléments en interactions, et à l'étude de la relation local-global.

Synthèse

La systémique, la cybernétique et les sciences de la complexité montrent que les méthodes analytiques sont trop restrictives pour décrire, concevoir et simuler le comportement d'un système complexe. Parmi les paradigmes introduits par ces communautés, deux concepts majeurs font l'unanimité :

1. les interactions entre les éléments du système doivent être utilisées,
2. le comportement des éléments doit être autonome.

Idéalement, la simulation des systèmes complexes nécessite la prise en compte de plusieurs problématiques [59], telles que la complexité structurelle (*self-relate*, environnement propre à chacun), la complexité comportementale (*self-regulate*, décision propre à chacun), ainsi que la complexité liée à la compréhension et à l'explication (*self-product*, le système est plus que la somme de ses parties).

3.2 Vers des approches informatiques

Grâce à ses capacités de calcul grandissantes et à son évolution rapide, l'informatique offre aujourd'hui des moyens de plus en plus intéressants pour illustrer les concepts développés par les sciences de la complexité, en offrant notamment la possibilité de simuler le comportement d'un grand nombre d'éléments en interaction. En réalisant des simulations en physique, chimie ou biologie. On arrive notamment à mieux comprendre et à enrichir notre connaissance sur certains phénomènes complexes.

Mais en même temps, la simulation de ces phénomènes exige aussi la modélisation et la conception de systèmes informatiques, qui deviennent eux aussi complexes. De ce fait, l'informatique tend de plus en plus vers le développement de nouvelles approches intégrant des capacités d'adaption et d'auto-organisation, tels que décrit par exemple par le Technical Forum "Self-Organisation in MAS" [25] ou l'*autonomic computing* [45] [73].

L'auto-organisation [25] Mécanisme ou processus qui permet à un système de changer son organisation pendant son exécution et sans contrôle explicite externe. Dans un système multi-agent l'auto-organisation se traduit par un changement de l'organisation entre les agents.

3.2.1 Les caractéristiques des nouvelles applications

Le développement de nouvelles applications informatiques dans lesquelles on cherche à promouvoir des capacités d'adaptation et d'autonomie ont naturellement des spécifications incomplètes. La plupart du temps, ces applications informatiques doivent répondre à plusieurs exigences parmi les suivantes :

- le système doit évoluer dans un environnement dynamique ;
- le système est ouvert, i.e. la cardinalité de ses composants est variable ;
- le système produit une fonction non-linéaire, i.e. dont une légère variation en entrée peut produire une grande variation en sortie du système ;
- le système comporte un nombre important de composants en interaction ;
- l'organisation interne du système est a priori inconnue ;
- le contrôle du système ne peut être centralisé.

L'introduction de ces nouveaux concepts nécessite de nouvelles approches informatiques, dont l'élaboration semble naturellement s'inspirer des sciences de la complexité. Ainsi depuis quelques années, la vie artificielle a utilisé de telles notions théoriques, qui ont permis la simulation et la compréhension de phénomènes thermodynamiques, physiques, sociologiques ou biologiques.

3.2.2 L'adaptation et l'apprentissage par auto-organisation

Pour répondre aux différents objectifs, l'utilisation de l'auto-organisation est un moyen intéressant pour concevoir ces systèmes [66]. En modifiant son organisation interne, un système auto-organisateur produit une fonction différente. Ce changement dû à la pression environnementale et aux interactions des éléments du système se fait pas à pas et de manière non supervisée.

D'autre part pour que l'auto-organisation du système soit effective, elle doit être basée sur l'apprentissage autonome de ses éléments. Cet apprentissage distribué et guidé par un ensemble de décisions locales s'affine en permanence sous l'influence de son environnement, grâce à la prise en compte des feedbacks environnementaux tel que décrit en section 3.1.2. Cette adaptation progressive correspond à un mode d'apprentissage comme on peut l'observer dans les réseaux de neurones [54].

Cependant et contrairement aux méthodes classiques d'apprentissage, l'adaptation du système par auto-organisation n'est pas fondée sur une évaluation globale du système, telle que la sélection des "meilleurs" individus ou bien par la connaissance de l'erreur entre la sortie effective et la sortie théorique dans un réseau de neurones [66].

L'intérêt de tels mécanismes d'auto-organisation repose sur les propriétés qu'il confère au système, à savoir des propriétés d'ouverture, d'émergence, d'adaptation et de robustesse.

3.3 Les systèmes multi-agents adaptatifs

Si l'on considère une définition très large, nous pouvons dire qu'un SMA est un système artificiel dont le fonctionnement repose sur l'interaction de ses agents. Le système est donc décomposé en sous-parties autonomes et actives, ce qui est donc bien adapté pour mettre en oeuvre les mécanismes d'adaptation et d'auto-organisation décrits précédemment.

3.3.1 La notion de système multi-agent

Dans les systèmes multi-agents auxquels nous nous intéressons, seuls les agents et les interactions liant les agents doivent être conçus. Ils sont munis de capacité de communication et d'un raisonnement qui collectivement leur permet de coopérer. Ensuite chaque agent utilise ses capacités ainsi que l'ensemble des degrés de liberté qui lui sont conférés pour favoriser l'émergence d'un fonctionnement adéquat pour le système. Pour que le comportement du système soit adéquat, les agents utilisent donc leurs capacités individuelles et leur perception environnementale, l'objectif étant pour chaque agent de mener une activité qui soit collectivement cohérente.

Ainsi, un système multi-agent auquel nous nous intéressons peut être défini comme un macro-système composé d'agents autonomes qui interagissent dans un environnement commun pour réaliser une activité collective cohérente, tout en négociant/gérant les objectifs individuels et conflits d'intérêts.

Dans ce type de système, le comportement global attendu est de réaliser des tâches données ou plus généralement de faire de la résolution de problèmes. L'objectif est alors d'exhiber un comportement global du système attendu par l'utilisateur en fonction des interactions entre les parties du système, ainsi qu'entre les parties du système et son environnement.

3.3.2 La notion de coopération

De nombreux travaux sur la coopération ont montré l'intérêt d'une telle approche comme moyen d'optimisation des performances (collectives ou individuelles), basé des fonctions de satisfaction de coûts [66]. La coopération est aussi un moyen pour des agents d'accomplir des tâches impossibles pour un individu isolé [32] et donc obtenir des fonctionnalités de plus haut niveau, comme s'adapter ou apprendre [38]. Nous pouvons distinguer deux types de coopération : la coopération intentionnelle et la coopération réactive. La coopération intentionnelle est issue de l'intelligence artificielle distribuée, et implique des capacités cognitives de représentation du monde et de planification, comme dans les travaux initiateurs des systèmes multi-agents [30]. Un autre type de coopération, fortement inspiré de la nature et des insectes sociaux, se focalise sur la coopération d'agents réactifs répondant simplement aux stimuli de leur environnement comme dans les travaux de Deneubourg et al. [24].
La théorie des AMAS (Adaptive Multi-Agent Systems) est basée sur les principes d'une coopération intentionnelle.

3.3.3 La théorie des Adaptive Multi-Agent Systems (AMAS)

Le but de cette théorie est de permettre à un système, utilisant des critères de réorganisation locale au niveau de ses entités, d'atteindre l'adéquation fonctionnelle. Par « adéquation fonctionnelle », nous nous référons à un comportement global du système, qui soit adapté à la tâche pour laquelle il a été conçu. La spécificité de cette théorie réside dans le fait que la fonction globale du système n'est pas implémentée dans les agents. Elle doit par conséquent émerger de leur comportement collectif. Pour vérifier cette adéquation, le comportement doit être jugé par un observateur extérieur au système qui connaît sa finalité. Pour garantir l'existence de ces systèmes coopératifs, la théorie des AMAS se base sur le théorème suivant démontré dans [37] :

Théorème 1 *Pour tout système fonctionnellement adéquat, il existe au moins un système à milieu intérieur coopératif qui réalise une fonction équivalente dans le même environnement.*

Un système possède un «milieu intérieur coopératif» lorsqu'il n'existe plus en son sein de situations antinomiques c'est-à-dire : d'incompréhension, d'ambiguïté, d'incompétence, de conflit, de concurrence ou d'inutilité entre les entités composant ce système. La démonstration de ce théorème repose sur un axiome et quatre lemmes. Notre but n'est pas de refaire cette démonstration en détail, mais de présenter succinctement cet axiome, ces quatre lemmes et leurs conséquences.

Axiome 1 *Un système fonctionnellement adéquat n'a aucune activité antinomique sur son environnement. Une activité antinomique sur l'environnement est une activité allant à l'encontre des intérêts de l'environnement.*

Lemme 1 *Tout système coopératif est fonctionnellement adéquat. Car par définition, un système coopératif n'a pas d'activité antinomique ou indifférente. Donc l'ensemble des systèmes coopératifs est inclus dans l'ensemble des systèmes fonctionnellement adéquats.*

Lemme 2 *Pour tout système S fonctionnellement adéquat, il existe au moins un système coopératif S* qui soit fonctionnellement adéquat dans le même environnement.*

Ce lemme lève l'incertitude restante vis-à-vis de l'existence d'un système coopératif équivalent pour chaque système fonctionnellement adéquat dans le même environnement. Sans ce lemme, l'intérêt porté aux systèmes coopératifs serait amoindri puisque pour certains systèmes fonctionnellement adéquats, il ne serait plus possible de trouver un système coopératif équivalent.

Lemme 3 *Tout système à milieu intérieur coopératif est un système coopératif.*

Donc l'ensemble des systèmes à milieu intérieur coopératif est inclus dans l'ensemble des systèmes coopératifs. Il reste donc comme précédemment, à s'assurer que chaque système coopératif dispose d'un système à milieu coopératif équivalent.

Lemme 4 *Pour tout système coopératif, il existe au moins un système à milieu intérieur coopératif avec une fonction équivalente dans le même environnement.*

La dernière incertitude étant levée, il est alors possible de définir une application surjective de l'ensemble des systèmes fonctionnellement adéquats vers l'ensemble des systèmes à milieu intérieur coopératif. Ces derniers systèmes constituant un sous-ensemble de l'ensemble des systèmes fonctionnellement adéquats.

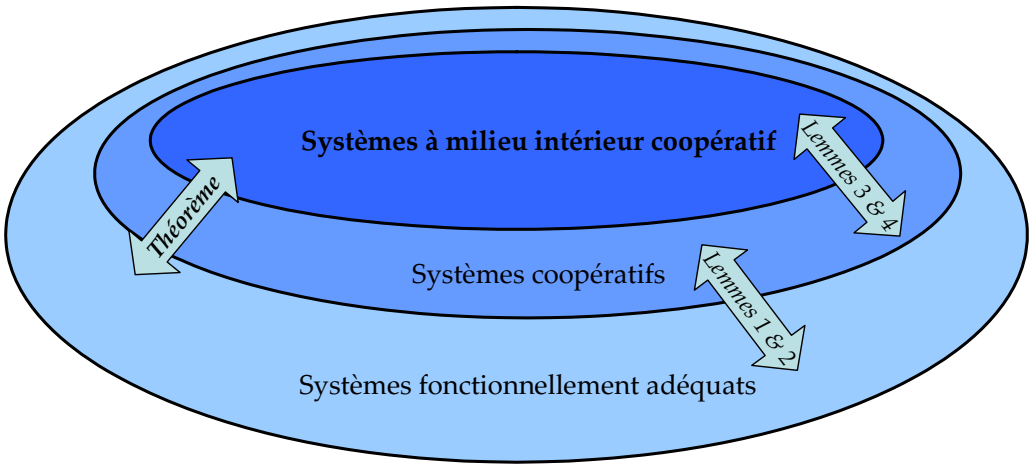


FIG. 3.1 – Relation entre les systèmes fonctionnellement adéquats et les systèmes coopératifs

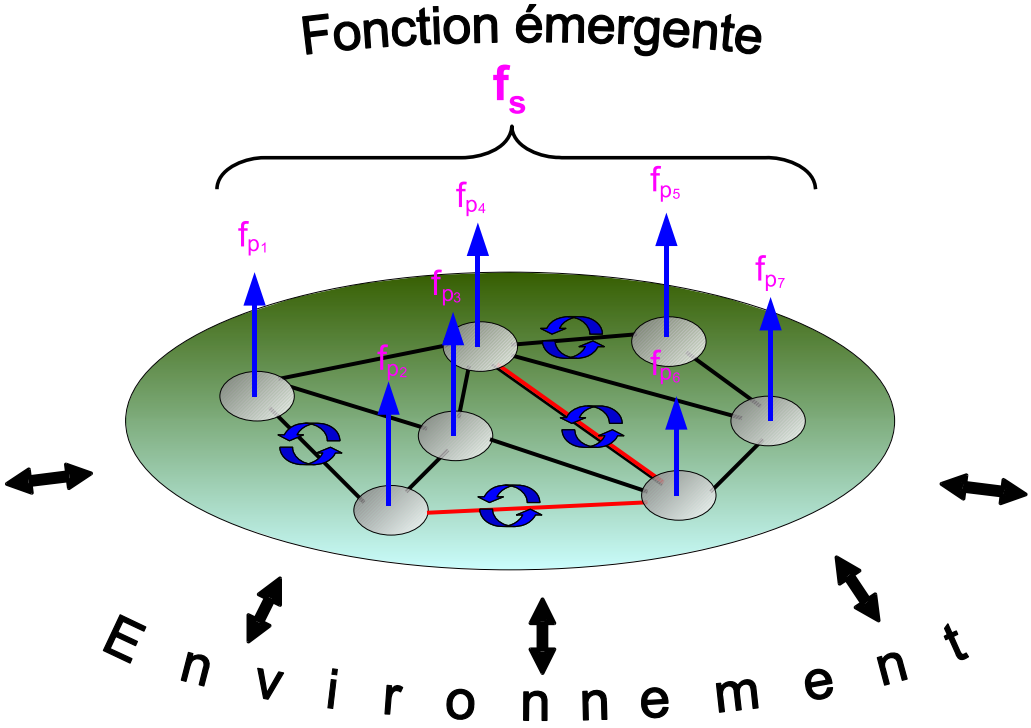


FIG. 3.2 – Adaptation par auto-organisation

Ainsi, le théorème central de la théorie des AMAS permet de se focaliser uniquement sur une classe de systèmes pour obtenir des systèmes fonctionnellement adéquats dans un environnement donné (voir figure 3.2). Le milieu intérieur coopératif d'un système est la notion centrale de la théorie des AMAS énoncée dans [15]. Cette théorie s'intéresse donc en particulier aux relations entre les entités constituantes des systèmes. Il devient alors clair qu'elle est aisément applicable aux systèmes multi-agents. La figure 3.2 présente comment doit s'adapter un tel système [34].

Le concepteur s'intéresse uniquement aux agents P_i qui produisent chacun leur fonction partielle f_{P_i} et leur donne le moyen de décider de changer les liens les unissant. Ce sont ces liens qui règlent la combinaison des fonctions f_{P_i} engendrant la fonction globale f_S. Ainsi, en fonction des interactions du système avec son environnement, l'organisation des agents se modifie pour faire face aux perturbations de l'environnement. Le système s'adapte à tout moment pour produire une nouvelle fonction f_S.

Finalement, la théorie des AMAS consiste en la définition pour chaque agent pris isolément, de tous ses états non coopératifs potentiels et de toutes ses activités permettant de les anticiper/détecter/supprimer. Il en résulte que la composition des fonctions partielles réalisées par chaque agent transforme la fonction globale du système en supprimant progressivement les états non coopératifs rencontrés localement.

3.4 Une théorie appliquée aux systèmes multi-agents

Pour pouvoir appliquer cette théorie, une spécification générale de l'architecture d'un agent AMAS a été proposée [8]. Un tel agent dispose en général des éléments suivants :

- des connaissances d'un domaine particulier permettant à l'agent de réaliser sa fonction partielle (compétences) ;
- d'une représentation de lui-même, des autres et de l'environnement (représentations) ;
- d'une attitude sociale coopérative, autrement dit de critères locaux lui permettant de savoir s'il est ou non coopératif et d'agir en conséquence (coopération) ;
- d'un langage d'interaction par envois de messages directs ou indirects par l'environnement (interaction) ;
- de capacités à raisonner sur ses connaissances et ses représentations (aptitudes) ;
- d'un milieu physique pour s'exprimer.

L'élément essentiel de cette architecture est l'attitude sociale coopérative de l'agent, qui doit anticiper, détecter et réparer une situation qu'il considère comme non coopérative. Cette attitude sociale définit donc des critères locaux qui vont permettre à l'agent de décider de son comportement. Cette logique a amené à la définition de règles permettant

la détection des situations non coopératives (SNC), exprimables en utilisant la logique suivante [15] :

$$\textbf{Non Cooperation} = \neg C_{per} \vee \neg C_{dec} \vee \neg C_{act} \tag{3.1}$$

L'agent peut donc percevoir trois types de situations non coopératives, telles que :

- $\neg C_{per}$ Le défaut de perception : lorsqu'un signal est incompris ou comporte des ambiguïtés. Pour y remédier l'agent peut par exemple tenter de transmettre l'information à d'autres agents potentiellement plus compétents.
- $\neg C_{dec}$ Le défaut de décision : l'agent est incapable d'exploiter un signal reçu. Il se considère improductif pour son environnement.
- $\neg C_{act}$ Le défaut de l'action : l'agent considère que son action est inutile à l'amélioration du collectif, ou qu'elle est conflictuelle/concurrente avec celle d'un autre agent.

En utilisant ces SNCs, l'agent cherche à modifier ses relations avec les autres afin de minimiser les SNCs. En modifiant son comportement, il ajuste sa fonction partielle, ce qui change la fonction globale. Ainsi, un agent réalise sa fonction partielle, tout en agissant pour l'organisation du collectif par la résolution des SNCs. Pour assurer l'émergence ainsi que l'adaptation du système, ces décisions sur la coopération doivent être locales aux agents et ne doivent pas être dictées par la fonction globale, qui émerge au cours du temps, en fonction des retours d'expérience de chaque agent et de la gestion des SNC.

Cette théorie a été expérimentée sur plusieurs cas d'applications et a conduit au développement de système, tels que :

- FORSIC [36] est un système de recherche d'informations dans un cadre éducatif. À partir des formateurs inscrits, le système a pour but de répondre aux requêtes d'un utilisateur en cherchant des cours dans un domaine donné. L'intérêt d'une approche AMAS dans ce cas est la possibilité de tenir compte de la dynamique des besoins des utilisateurs et des services offerts.
- ABROSE [35] est un système de courtage électronique de services sur internet. Chaque agent représente un fournisseur de services ou un consommateur de services. Ils travaillent sur leurs accointances exploitées pour faire suivre les requêtes utilisateur à des agents jugés plus compétents lorsqu'ils ne peuvent pas satisfaire complètement les besoins.
- STAFF [67] est un système de prévision des crues. Il s'agit là d'un problème complexe et dynamique composé de nombreux paramètres. Dans le système, chaque agent est responsable d'une station et effectue un réajustement permanent relativement à ses mesures d'entrée, en interaction avec les résultats des autres agents.
- SYNAMEC [16] est un projet de conception mécanique assisté par un AMAS. Chaque agent représente une sous-partie du mécanisme à construire. En interac-

tion avec les autres, il cherche la position idéale et la dimension idéale pour réaliser une trajectoire.

– DYNAMO [61] est un système d'aide à la construction d'ontologie. Chaque agent représente un concept. En interagissant avec les autres et en utilisant des algorithmes distribués d'analyse statistique, ils cherchent collectivement à trouver la structure ontologique la plus satisfaisante à l'analyse syntaxique.

Les résultats obtenus par ces systèmes ont montré la pertinence de l'approche par AMAS pour la résolution de problèmes complexes, et ont conduit à l'introduction de la méthode ADELFE [9], qui fournit un guide pour la construction de ce type de système. Cette méthode étend des méthodes objets telles que le cycle de développement RUP (*Rational Unified Process*) et le langage UML (*Unified Modeling Language*), afin de répondre aux besoins spécifiques de la théorie des AMAS. Pour une description détaillée de la méthodologie, on peut consulter le site du projet : www.irit.fr/Adelfe.

Synthèse : les objectifs de la thèse

Dans cette partie, nous avons vu les différentes techniques existantes et la manière d'aborder la conception avion. Les approches traditionnellement utilisées favorisent la description d'un problème par une formulation mathématique pour pouvoir utiliser des approches d'optimisation classiques. Cette manière complètement déterministe d'aborder le problème s'explique par le manque d'outils et d'approches permettant de modéliser certaines des caractéristiques de la conception avion, telles que :

- les nombreuses interactions/interdépendances entre les disciplines (complexité structurelle) ;
- la non-linéarité comportementale observée à la suite d'une modification de paramètres de conception (complexité comportementale) ;
- l'impossibilité d'exprimer des règles de régulation adaptées à toutes les situations (complexité décisionnelle) ;

L'évolution récente de l'informatique vers des approches auto-organisatrices et adaptatives permet aujourd'hui d'envisager d'autres manières d'aborder ce type de problème, en le considérant comme un système complexe. En utilisant la dynamique du système, ces approches s'adaptent et apprennent les propriétés intrinsèques du système.
Cette nouvelle manière de voir les choses permet de prendre en compte simultanément plus de paramètres et surtout d'aller vers des approches transparentes du point du vue du comportement du système. Ainsi, en augmentant le nombre de paramètres que l'on peut considérer, on s'offre de nouvelles possibilités en termes de parcours de l'espace de recherche et une meilleure visibilité des conflits apparents.

Les systèmes multi-agents adaptatifs (AMAS) ont montré des capacités pour modéliser des systèmes complexes, nous proposons de les utiliser dans un système résolvant des problèmes de conception préliminaire avion, et d'illustrer les nouvelles solutions qui sont offertes aux concepteurs.

Les objectifs industriels

D'un point de vue industriel, l'objectif de la thèse est de fournir de nouveaux outils pour résoudre des problèmes difficiles de conception, pour lesquels les objectifs sont nombreux et interdépendants. Pour développer ce genre d'outils, nous proposons d'utiliser les SMAs et de démontrer qu'ils sont utilisables dans un contexte d'assistance à la conception de produits ou de systèmes complexes.

Un de nos objectifs est donc aussi de justifier ou non l'introduction des SMAs en milieu

industriel en :

- identifiant les apports des SMAs à la résolution d'un problème de conception avion, en le comparant aux outils existants ;
- réalisant un démonstrateur qui illustre l'utilisation et les apports du SMA, pour un concepteur.

Les objectifs théoriques

Pour proposer une approche SMA qui permette de réaliser de tels outils, il est nécessaire de faire évoluer l'optimisation distribuée par recherche locale, en proposant une approche qui puisse intégrer les besoins et objectifs de chacune de ces parties.

Nous proposons de traiter ce problème en utilisant une approche à base d'AMAS. Cette théorie a été utilisée pour réaliser différents outils de résolution de problème, mais jamais dans le cadre de problèmes d'optimisation qui sont multidisciplinaires et multi-objectifs.

La démarche consiste donc à étendre la théorie des AMAS à notre problème et à définir un algorithme de résolution, qui traite des contraintes interdépendantes, conflictuelles et réparties au sein d'un collectif d'agents. Ceci revient principalement à définir les agents et leurs rôles, à identifier leurs situations non-coopératives et à définir des mécanismes de coordination et de communication.

Dans cette thèse, il s'agira aussi de montrer si la problématique de l'auto-adaptation permet désormais d'élaborer des systèmes efficaces et industrialisables, en les comparant à d'autres techniques.

Les principes de l'approche

Dans notre approche, un agent est associé à chaque modèle disciplinaire impliqué dans la fonction d'évaluation (figure 3). En utilisant leurs capacités de coopération, ces agents auto-ajustent les paramètres de la fonction avion dans le but de converger vers un état adéquat. Ce comportement adéquat est composé d'un vecteur de paramètres de conception, permettant de réaliser l'ensemble des performances attendues par le concepteur avion. Finalement en définissant les connaissances propres à chaque agent et en utilisant un raisonnement coopératif, le collectif s'organise en apprenant les interdépendances implicites et détermine ainsi un ensemble de valeurs admissibles et adaptées au problème.

La résolution par AMAS est située dans l'interaction entre les agents (micro-niveau). Contrairement aux approches classiques, elle ne se situe donc pas au niveau d'un mécanisme qui raisonne sur l'espace de recherche global (macro-niveau). Un solveur AMAS est transparent, car les connaissances métiers sont toujours explicitement observables durant une résolution. Notre objectif est que la résolution émergente de problèmes favorise un

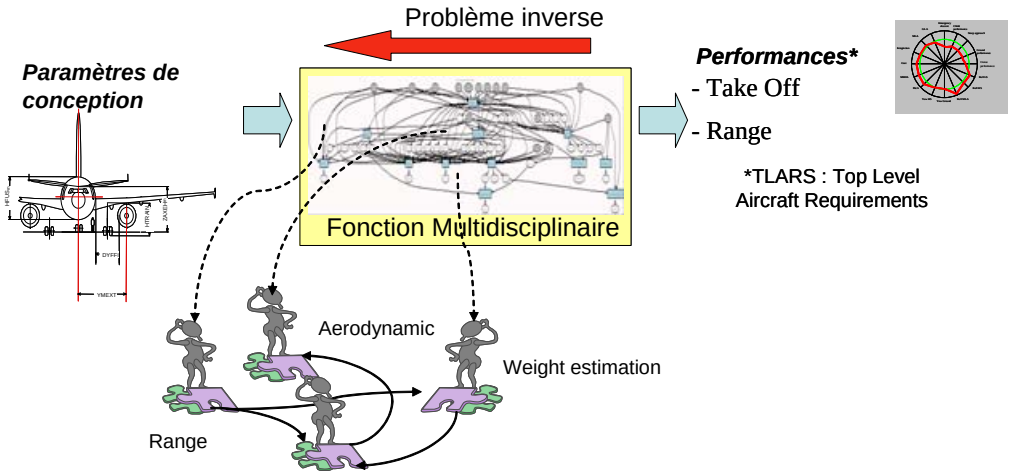


FIG. 3 – La conception avion et les agents disciplines

processus de co-construction en temps réel avec les concepteurs.

Deuxième partie

Les contributions

Introduction

La construction de notre solution est organisée selon plusieurs axes :

- L'axe principal est la proposition d'un système multi-agent permettant d'effectuer la résolution d'un problème de satisfaction de contraintes dans un système multi-disciplinaire. Cette première étape, nous a permis de définir plusieurs types d'agent, de leur associer un raisonnement coopératif et de faire de la résolution de problème par auto-régulation. Cette architecture d'agents définie dans le chapitre 4, régule des conflits par auto-organisation et aboutit à la convergence vers un état satisfaisant l'ensemble des contraintes associés aux modèles disciplinaires, lorsque cela est possible.
- Ce premier axe résout des problèmes de satisfaction de contraintes sans favoriser la recherche d'une solution particulière. Afin de parcourir le front de Pareto (introduit en section 2.1.2), le second aspect de notre recherche consiste à piloter le système d'auto-régulation de contraintes pour favoriser l'optimisation de certains critères. Ce parcours de solutions participe à la compréhension du problème multi-objectif, qui nécessite la mise à disposition d'un ensemble de solutions à interpréter et à argumenter. Le chapitre 5 présente donc un modèle d'optimisation capable de parcourir des fronts de Pareto en pilotant notre modèle d'auto-régulation.
- Le troisième axe aborde la conception avion comme un processus de co-construction itératif, dans lequel la vision du concepteur est un élément clé. L'utilisation d'un système multi-agent pour résoudre ce problème offre une nouvelle façon de le traiter qui nécessite la prise en compte de nouvelles connaissances et de nouveaux besoins pour le concepteur en termes de compréhension et de pilotage du système. Des solutions pour éclairer le concepteur sur le mode de fonctionnement du SMA sont proposées dans le chapitre 6 à travers de nouveaux moyens d'interactions et de pilotage.

L'organisation de cette partie est résumée en figure 4.

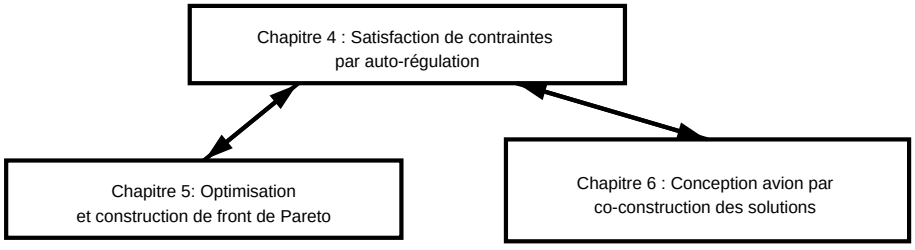


FIG. 4 – Organisation des chapitres de la seconde partie

La connaissance s'acquiert par l'expérience, tout le reste n'est que de l'information.

Albert Einstein

4

Un SMA adaptatif pour l'auto-régulation sous contraintes

Sommaire

En conception avion, un concepteur choisit les paramètres pour lesquels il a des objectifs. Ensuite, il définit les contraintes principales sur la conception et les performances de l'avion qui sont attendues. Pour trouver un avion qui réponde aux objectifs qu'il s'est fixé, le concepteur construit alors une fonction de simulation composée de modèles disciplinaires, tels que l'estimation de la masse ou le calcul de performance aérodynamique, etc. En général, les contraintes de conception sont des entrées globales de cette fonction et les performances des sorties. Mais un avion est un produit complexe, de ce fait la plupart des objectifs de conception attendus sont interdépendants les uns des autres et la fonction de simulation s'apparente à un système complexe composé d'interdépendances et de boucles de rétroactions. Par exemple, certains paramètres de la simulation sont souvent partagés par plusieurs disciplines ayant des objectifs communs ou conflictuels. Au niveau de la structure du problème, les boucles se traduisent par la présence de paramètres qui sont à la fois des entrées (degrés de libertés) et des sorties (performances) du système. Finalement, la conception préliminaire d'un avion revient donc à définir une fonction de simulation dont les propriétés sont celles d'un système complexe. Cette complexité a plusieurs origines :

- *La complexité structurelle* provient du nombre de composants à utiliser pour mettre en place une simulation. Dans notre cas, ces composants sont des modèles de calculs disciplinaires, permettant l'évaluation de performance de l'avion. La complexité de notre structure provient donc aussi de *la multitude de liens* unissant ces composants.
- Le fonctionnement du système produit *des comportements qui ne sont pas facilement déductibles* conduisant à une *complexité décisionnelle*. Par exemple, à cause des interdépendances entre les disciplines, le système peut entrer dans des raisonnements chaotiques. En particulier, des modifications sur des paramètres de bas niveau peuvent avoir des conséquences importantes sur l'évolution du comportement global. Cette complexité est difficilement maîtrisable et c'est pourquoi il est nécessaire de l'étudier.

Dans ce chapitre, nous proposons un système multi-agent adaptatif pour l'auto-régulation de contraintes dans un système multi-disciplinaire et multi-objectif. Notre système est composé de quatre types d'agents (boucle de rétroactions, objectif d'entrée, objectif de sortie et disciplinaire). Ils sont pourvus de connaissances locales et ont une représentation de leur environnement. Ils prennent leurs décisions en utilisant ces connaissances, une attitude sociale coopérative ainsi que des capacités d'adaptation.
Ainsi dans les sections suivantes nous présenterons les connaissances des agents et leur raisonnement.

4.1 La formulation du problème et son agentification

Un système complexe est composé de boucles de rétroactions et d'interdépendances, qui doivent être maîtrisés pour permettre au système de converger. Pour s'affranchir du problème de boucles, les algorithmes de DisCSP transforment la formulation du problème en un arbre DFS (section 2.3). Cet arbre de décomposition permet de vérifier l'exactitude des solutions trouvées en introduisant un ordre dans la résolution du système. Mais tel que décrit en 2.3.6, ce type d'approche est incompatible avec la structure de notre problème. En MDO, l'égalité des boucles (boucles de rétroactions) est permise soit par l'utilisation de solveurs, soit par l'intégration d'optimiseurs qui ont des contraintes sur la réalisation d'égalités entre les entrées et les sorties. Dans le premier cas, le système est cohérent à chaque évaluation et dans le second uniquement lorsque l'optimiseur garantit l'égalité des contraintes.

- Dans notre approche, les boucles du système sont gérées en utilisant des *agents boucles de rétroactions* (AB), dont le rôle est d'égaler la valeur des paramètres qui sont à la fois en entrée et en sortie du système. Concrètement, leur objectif est de faire converger les valeurs fournies au système ainsi que celles calculées vers une même valeur.
- L'intégration des disciplines est assurée par les *agents disciplinaires* (AD). Une organisation de ces ADs constitue un réseau de connaissances et de dépendances du domaine à simuler. Dans cette organisation, chaque AD a des capacités d'exécution, des connaissances métiers, ainsi que des compétences pour communiquer et adapter son comportement.
- Les *agents objectifs d'entrée* (AE) sont créés pour chaque paramètre d'entrée du système qui est partagé par plusieurs disciplines. Leur rôle est de décider de la valeur du paramètre partagé, en analysant les préférences des disciplines et en choisissant un compromis.
- Enfin les *agents objectifs de sortie* (AS) gèrent les valeurs des objectifs de performances attendues par le concepteur. Pendant les itérations du système, leur rôle est de vérifier que les valeurs de sortie calculées par les agents disciplinaires respectent bien les objectifs attendus.

La figure 4.1 montre l'agentification d'un système bouclé, composé de deux modèles disciplinaires.

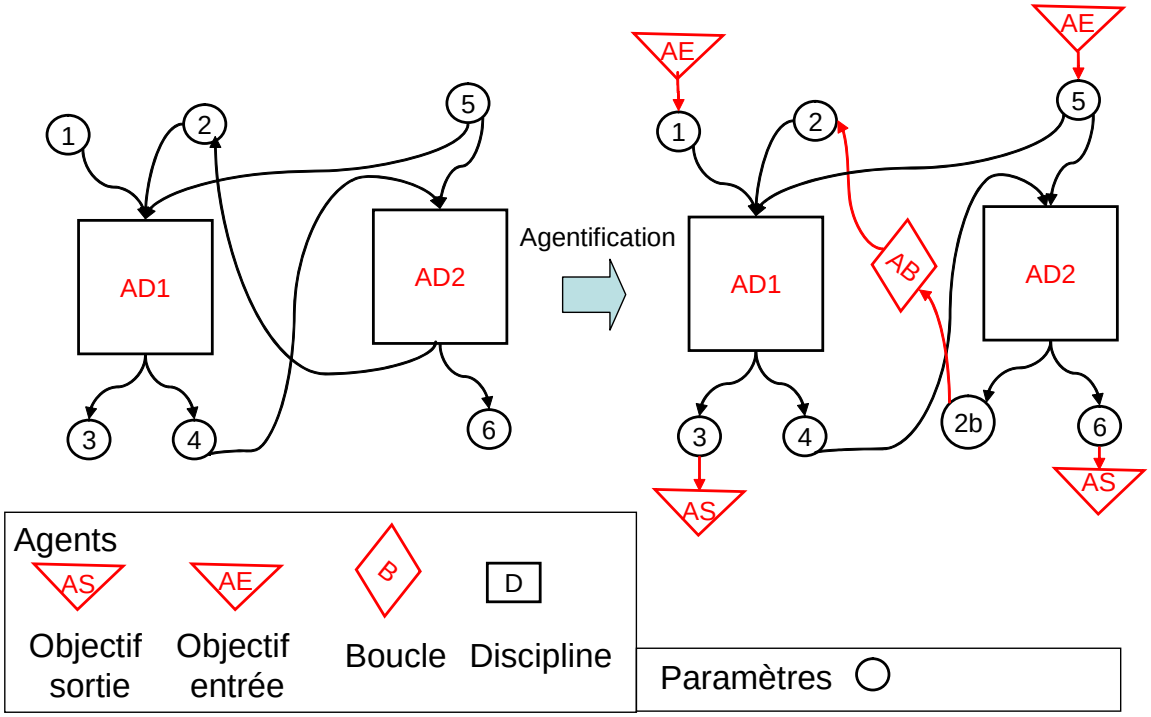


FIG. 4.1 – Agentification d'un problème à deux disciplines

4.2 Les choix de modélisation des agents

4.2.1 Les connaissances locales des agents

Dans des simulations multi-disciplinaires, chaque discipline est représentée par un modèle. Ce modèle permet, à partir d'hypothèses et de données reçues en entrée, de calculer des performances et des métriques relatives à la discipline modélisée. Pour s'assurer de la précision et de l'exactitude des calculs, des informations qualitatives sur les entrées des modèles sont à préciser. Elles permettent, par exemple, de tenir compte des hypothèses selon lesquelles le modèle a été conçu. Pour prendre en compte ces connaissances, nous avons choisi un modèle générique, qui permet de qualifier la valeur d'une entrée. Il tient compte de domaines de validité définis sous la forme d'intervalles.

Les domaines de validité

Pour chaque entrée x_i d'un agent A_q, on indique des domaines de validité à l'aide d'intervalles :

– Un premier intervalle, noté $D_{Phys}^{Aq,xi}$ donne les limites inférieures et supérieures des variables de conception pour chaque entrée du modèle, à l'intérieur desquelles le modèle est calculable. Il s'agit donc d'un intervalle fournissant les limites physiques

de fonctionnement du modèle.

– Un second intervalle de validité, noté $D_{Obj}^{Aq,xi}$ décrit un intervalle objectif préféré pour le paramètre. Ainsi, toutes valeurs situées à l'intérieur de cet intervalle respectent les contraintes fixées par le concepteur.

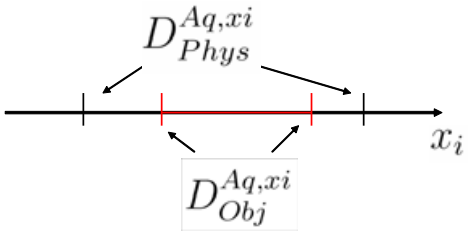


FIG. 4.2 – Intervalles objectifs

La mesure de la criticité

En utilisant ces intervalles, une fonction mathématique, notée $F_{C_r}^{Aq,xi}$ continue par morceaux est construite pour chaque entrée x_i d'un agent A_q (figure 4.3). Elle permet à l'agent de calculer un critère de non-satisfaction pour chacune de ses entrées. Ce critère, aussi appelé criticité C_r, tient compte des limites physiques et des objectifs du paramètre. Chaque fonction $F_{C_r}^{Aq,xi}$ est construite de la manière suivante et en fonction de la valeur de x_i :

$$\forall x_i \quad \exists \quad F_{C_r}^{Aq,xi} \tag{4.1}$$

$$\forall x_i \quad \in D_{Obj}^{Aq,xi} \quad -1 \leq F_{C_r}^{Aq,xi} \leq 0 \tag{4.2}$$

$$\forall x_i \quad \notin D_{Obj}^{Aq,xi}, \in D_{Phys}^{Aq,xi} \quad 0 < F_{C_r}^{Aq,xi} < \textit{Seuil maximal de Criticité} \tag{4.3}$$

$$\forall x_i \quad \notin D_{Phys}^{Aq,xi} \quad F_{C_r}^{Aq,xi} = \textit{Seuil maximal de Criticité} \tag{4.4}$$

Ainsi, la criticité d'un paramètre prend les valeurs suivantes :

– quand la valeur de l'entrée est à l'intérieur de l'intervalle objectif, sa valeur critique (*criticité*) est négative ;

– quand la valeur de l'entrée est à l'intérieur de l'intervalle physique mais en dehors de l'intervalle objectif, alors sa criticité est positive mais inférieure au seuil de criticité maximale, prédéfini par le concepteur du modèle.

– enfin, quand la valeur de l'entrée est en dehors des limites physiques, alors la valeur critique associée est égale au seuil maximal de criticité.

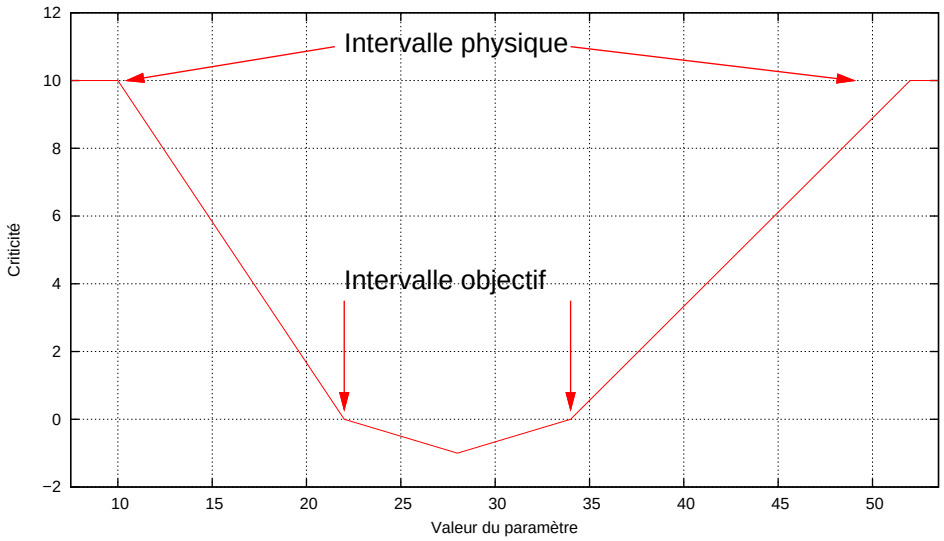


FIG. 4.3 – Fonction d'évaluation des criticités

La criticité des agents

Pour négocier, comprendre les besoins de leur environnement, mesurer leur degré de satisfaction, les agents ont besoin d'évaluer leur état, en calculant leur *criticité*.

- Concernant les agents disciplinaires (AD), leur niveau de satisfaction doit être fortement lié à celui de leur modèle disciplinaire. Ainsi la criticité d'un AD correspond à la criticité maximale parmi celles de ses entrées. Concrètement, un agent à 3 entrées ayant comme valeurs de criticités respectives 50, 60, 250 a une criticité de 250.
- Les agents objectifs d'entrée (AE) et de sortie (AS) gèrent un seul paramètre à la fois. Leur criticité est donc simplement celle de leur paramètre. Pour mesurer la criticité de ce paramètre, deux intervalles sont donc définis et une fonction $F_{C_r}^{Aq,xi}$ associée. Dans ce cas, l'intervalle physique représente alors l'ensemble des valeurs que l'agent pourra sélectionner, alors que l'intervalle objectif représente l'ensemble des valeurs qui satisferont les exigences du problème.
- Enfin, la criticité des agents boucles de rétroactions (AB) tient compte de deux aspects. Une première mesure vérifie que la valeur obtenue en sortie corresponde bien aux objectifs du problème, tel qu'un agent objectif le ferait. Une seconde mesure permet d'évaluer l'écart de valeurs obtenu entre la valeur fournie au système et la valeur calculée. La criticité de l'agent est alors simplement la criticité maximale parmi ces deux valeurs.

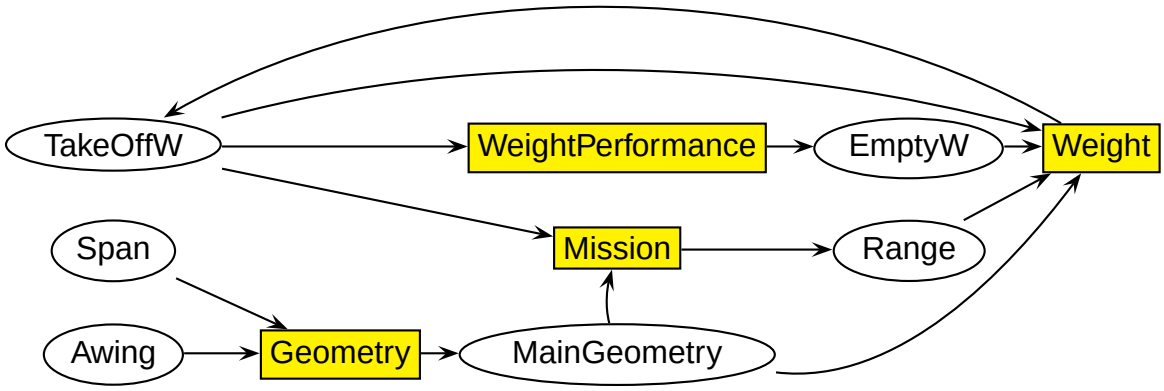


FIG. 4.4 – Un exemple simple de relations entre modèles disciplinaires

La gestion du modèle physique

Pour connaître leur valeur de sortie, les agents disciplinaires ont la capacité d'exécuter leur modèle. Ils peuvent aussi avoir le besoin d'effectuer d'autres opérations sur leur modèle telles que des résolutions inverses, en utilisant des méthodes d'optimisation appropriées. A l'aide de ces résolutions inverses, il estime les dépendances entre leurs entrées/sorties. Ainsi, ils peuvent trouver des moyens de modifier leurs paramètres d'entrée, pour répondre à des besoins de variation sur leurs sorties. Dans notre modèle, les agents utilisent simplement un calcul de gradient, via la matrice Jacobienne[3] du modèle, pour mesurer les variations.

Le pas de modification

Finalement, les ajustements qu'un agent peut effectuer sur ses entrées sont contraints par un pas de modification défini pour chacune d'entre elles. Ce pas de modification prend en compte la largeur de l'intervalle objectif du paramètre, et est modifié par l'agent en fonction de ses perceptions environnementales, tel que nous le verrons à la section 4.3.3.

4.2.2 Les connaissances environnementales de l'agent

Pour interagir, chaque agent a besoin de connaître ses relations et collaborations directes. Dans notre cas, ce sont ses agents appelés fournisseurs et consommateurs de paramètres. L'exemple présenté en figure 4.4 illustre quelques dépendances disciplinaires simplifiées de la conception avion. Dans le sens direct, un agent consommateur reçoit la valeur calculée par son fournisseur. Par exemple, le modèle *Weight* est consommateur

[3]Matrice des dérivées partielles : la dérivée partielle d'une fonction est la dérivée par rapport à l'une de ses variables, les autres étant gardées constantes

des paramètres *Range*, $EmptyW$, et $MainGeometry$. Dans le sens indirect lorsque des modifications sont nécessaires, l'agent consommateur formule ses besoins et contraintes et les envoie à son fournisseur. Généralement, les agents disciplinaires jouent à la fois les rôles de fournisseurs et de consommateurs, puisqu'ils reçoivent et fournissent des valeurs de paramètre. Par exemple, le modèle $Weight$ est aussi fournisseur du paramètre $TakeOffW$. Les agents objectifs d'entrée, quant à eux, sont uniquement fournisseurs, puisqu'ils décident de la valeur d'un paramètre d'entrée du système. Enfin, les agents objectifs de sortie sont des consommateurs, puisqu'ils reçoivent une valeur de sortie du système.
Finalement, l'agent a donc des connaissances relatives à son environnement direct, puisqu'il connaît ses collaborateurs. Cependant, cette connaissance reste locale, car il n'a pas de connaissance plus globale, ni sur la topologie du système, ni sur les agents présents au-delà de son voisinage direct.

4.2.3 Le comportement coopératif et le rôle des agents

Munis de connaissances environnementales et locales, les agents ont la capacité de se situer dans leur environnement. En utilisant ces connaissances et un modèle de décision, ils peuvent faire des choix pour collaborer avec leur entourage.
Dans notre architecture, nous utilisons un raisonnement coopératif basé sur la théorie des AMAS [17]. Dans cette section, nous présentons l'application des lignes directrices de ce raisonnement, en définissant le rôle du raisonnement coopératif et en illustrant quelques règles non coopératives.

Le comportement coopératif

Notre stratégie coopérative peut être énoncée comme suit : pour choisir l'action la plus coopérative, chaque agent compare la mesure de sa criticité avec celles communiquées par ses voisins. Ensuite, il agit pour réduire la criticité la plus mauvaise. En utilisant une mémoire de ses actions passées et un ensemble de règles coopératives [75], il choisit alors l'action qui lui permet d'améliorer l'état du paramètre qu'il perçoit comme étant le plus critique. Finalement, le but du raisonnement coopératif est d'effectuer l'action qui diminue la situation la plus critique dans le voisinage perçu par l'agent. Ainsi, il peut être résumé selon les principes suivants :

- Si l'agent estime qu'il est localement le plus critique, alors il construit une requête de modification à adresser à son prédécesseur selon ses propres besoins.
- Sinon si l'agent est moins critique qu'au moins une des requêtes de modification qu'il a reçues, alors il agit dans l'intérêt de son voisinage.

Le raisonnement coopératif suit donc les étapes suivantes :

1. L'agent sélectionne la requête la plus critique.
2. Il construit de nouvelles requêtes de modification, qui vont permettre de répondre à cette demande. Pour cela, l'agent détermine la matrice Jacobienne de son modèle et trouve ainsi les dépendances locales entre les demandes de modifications provenant de ses sorties et les modifications à effectuer sur ses entrées.
3. Enfin, lorsque l'entrée manipulée n'a aucun fournisseur, alors l'agent a la capacité d'adapter le paramètre en modifiant lui-même sa valeur. Lorsque l'entrée est fournie par un autre agent, il demande une modification du paramètre à son fournisseur, ce qui va permettre de satisfaire l'intérêt du consommateur le plus critique.

Le rôle des agents

Sur nos différents agents, ce raisonnement se traduit par la distribution des rôles et des moyens d'action suivants :

- Un *agent objectif de sortie (AS)* contrôle la valeur des objectifs de sortie. Son rôle est de s'assurer que les performances obtenues sont conformes avec les données du problème. Il ne possède aucun utilisateur de paramètre, et agit donc uniquement dans son intérêt. Lorsqu'il reçoit une valeur en provenance de son fournisseur, il utilise son intervalle objectif et construit une requête de modification qu'il transmet à son fournisseur.
- Un *agent objectif d'entrée (AE)* reçoit des demandes de modification en provenance de ses consommateurs. Son rôle est de modifier la valeur du paramètre d'entrée en fonction de l'ensemble des demandes qu'il reçoit. Pour choisir la modification à prendre en compte, il utilise le raisonnement coopératif. Ensuite en utilisant son pas d'adaptation (introduit en 4.2.1 et repris en 4.3.3), il modifie sa valeur et en informe ses consommateurs.
- Un *agent disciplinaire (AD)* reçoit des valeurs d'exécution en provenance de ses fournisseurs et des demandes de modification en provenance de ses consommateurs. Son rôle est de satisfaire ses objectifs et ceux de ses consommateurs en utilisant le raisonnement coopératif. Il a la capacité de sélectionner la requête qui lui semble la plus critique. En utilisant des algorithmes de résolution inverse, il est capable de trouver les interdépendances entre ses entrées/sorties, puis de transmettre des demandes de modifications à ses fournisseurs.
- Enfin le rôle principal de l'*agent boucle (AB) de rétroactions* est de vérifier que la différence entre la valeur de l'objectif fournie en entrée du système est négligeable avec celle obtenue en sortie. Mais il peut également jouer un second rôle celui d'*agent*

objectif de sortie, lorsque la valeur du paramètre de sortie dont il a la responsabilité doit également répondre à un objectif donné par le concepteur. Ainsi dans un cas, il vérifie que la solution est physiquement viable (cohérente), et dans l'autre il garantit que l'objectif souhaité par le concepteur est atteint.

4.2.4 La communication

Pour permettre aux agents d'interagir et de travailler ensemble, il est nécessaire d'établir une communication. Les travaux traitant de la communication dans un SMA, montrent qu'il existe deux modes de communication : les communications directes et indirectes. *La communication indirecte* est une démarche de communication, qui passe généralement par l'environnement. Ainsi, pour communiquer, les agents agissent sur leur environnement ; ce qui modifie la partie visible d'un ou de plusieurs agents à qui l'information doit être signifiée. Cette communication implique donc que les agents soient dans un environnement physique commun qui ne permet pas de communication avec un agent particulier.
A l'opposé, *une communication directe* est une communication, où l'agent s'adresse individuellement aux autres par l'envoi de messages. Ce type de communication est beaucoup plus riche et complètement adapté à des applications physiquement et conceptuellement distribuées. Dans ces applications, les agents utilisent des actes de langage, ont une connaissance de leurs voisinage, et on ne peut partager toutes les informations avec l'ensemble des agents. C'est cette approche que nous utilisons, car les systèmes que nous adressons sont conceptuellement distribués et s'échangent des flux d'informations. Ils ne permettent donc pas le partage d'un environnement global.

Une fois le mode d'échange d'information sélectionné, il faut définir les informations à échanger (type de message, contenu des messages). L'efficacité de la résolution d'un problème distribué dépend de la qualité, de la quantité et de la fréquence des informations échangées. En ajoutant et en partageant plus de connaissance et donc en améliorant la quantité et la qualité des informations reçues par les agents, on est censé améliorer leur perception environnementale et donc la qualité de leur décision.
Cependant, partager plus d'information, c'est forcément renoncer à d'autres choix en termes d'adaptation, d'autonomie et de distribution du système. En effet, dans certaines situations, augmenter la quantité d'information échangée se traduit aussi par une réduction de l'efficacité de la recherche.

Dans notre approche, nous avons cherché à limiter l'échange d'informations, à celles nécessaires à la résolution des situations non coopératives et à celles qui aident le système à converger. Deux types de messages cohabitent : ceux permettant à un agent d'envoyer à ses utilisateurs des valeurs de paramètres calculées sous la forme de messages d'exécution, et ceux permettant de demander à ses fournisseurs des valeurs souhaitées sous la forme

de requêtes de modification.
En exécution la structure du message est la suivante :

- Message Name : ForwardMessage
- Origine : <Identifiant de l'expéditeur>
- Destination : <Identifiant du destinataire>
- Liste de valeurs :
  - <Nom du paramètre>
  - <Valeur du paramètre>
- <Criticité de l'agent origine>

En modification :

- Message Name : BackwardMessage
- Origine : <Identifiant de l'expéditeur>
- Destination : <Identifiant du destinataire>
- Liste de valeurs :
  - <Nom du paramètre>
  - <Valeur du paramètre>
  - <Criticité du paramètre>
  - <Valeur du paramètre attendu>
  - <Criticité attendue>
  - <Agent initiateur de la modification>
  - <Paramètre de l'agent initiateur de la modification>
- <Criticité de l'agent origine>

Chaque message est accompagné d'informations qualitatives, comme le nom et l'unité de référence du paramètre, la valeur associée à la requête, la criticité de la demande, ou encore la criticité de l'initiateur de la requête. Dans les sections suivantes, nous justifions les informations contenues dans ces messages, en montrant comment elles sont exploitées par les agents.

4.2.5 La coordination

Le raisonnement coopératif présenté permet à l'agent d'agir en fonction de l'état de son environnement. Comme dans tout système réparti, des choix de coordination des éléments sont nécessaires. Lors de la définition d'une simulation classique, le concepteur définit en principe un *workflow* qui fournit une séquence d'exécution. Utiliser des agents autonomes, nous incite à chercher des modes de coordination basés sur les connaissances locales des agents.

Un moyen simple de coordonner les agents est de définir une séquence locale d'exécu-

tion. Dans ce cas, les agents agissent et communiquent avec leur environnent en respectant un protocole de coordination statique.

Par conséquent, en mode dit d'exécution, l'agent attend d'avoir reçu toutes les valeurs de ses entrées en provenance de ses fournisseurs avant de s'exécuter ; c'est-à-dire d'effectuer le calcul lié à son modèle.
En mode dit indirect, il attend d'avoir reçu l'ensemble des demandes en provenance de ses utilisateurs avant de générer ses propres demandes.
Cependant pour que le comportement global soit cohérent, il faut gérer les boucles contenues dans le système, ce qui justifie l'utilisation d'agent boucle de rétroactions et donc l'agentification de notre problème, telle qu'illustrée en section 4.1

Nous avons choisi ce mode de coordination pour sa simplicité, car il permet de s'affranchir de contraintes temporelles des agents. Mais un autre mode de coordination que nous avons expérimenté et basé sur la dynamique des agents sera discuté en perspectives, il permet d'adapter l'envoi de message en fonction des situations critiques rencontrées par chaque agent au cours de la résolution. Dans ce cas, la coordination des agents ne se fait plus en fonction de la structure du problème, mais en fonction des perceptions environnementales des agents.

4.2.6 Synthèse des choix de modélisation

Dans cette section, nous avons présenté les agents du système (AD, AE, AS, AB) dont une synthèse est proposée dans le tableau 4.1. Les principes du raisonnement coopératif, ainsi que des choix de coordination et d'organisation ont également été proposés. Mais jusqu'à présent, les agents agissent sans anticipation et localement. Or toutes les actions prises localement ont typiquement des effets multiples dans diverses parties du système et des conséquences globales.

Type d'agent	Rôles	Objectifs individuels	Actions	Propriétés physiques	Symbole
Disciplinaire (AD)	Satisfaire ses objectifs et ceux de ses consommateurs, en utilisant le raisonnement coopératif	Faire respecter les domaines de validité à l'ensemble de ses paramètres	Il choisit la modification qui lui semble comme étant la plus critique pour son environnement. Il transmet cette modification à ses fournisseurs, et adapte ses entrées libres de tout fournisseur	– Nb d'entrée : j [1..n] – Nb de sortie : k [1..n] – Nb de domaine de validité : j – Nb pas de modification : m <= j	
Objectif Entrée (AE)	Décider de la valeur d'un paramètre système partagé par plusieurs disciplines	Son objectif individuel est que son paramètre soit dans le domaine de validité défini par le concepteur	Pour modifier la valeur de son paramètre, il analyse les préférences des disciplines et choisit un compromis, en utilisant le raisonnement coopératif	– Nb d'entrée : 0 – Nb de sortie : 1 – Nb de domaine de validité : 1 – Nb pas de modification : 1	
Objectif Sortie (AS)	S'assurer que les performances obtenues sont conformes avec les données du problème. Il ne possède aucun utilisateur de paramètre, et agit donc uniquement dans son intérêt	Son objectif est que la valeur que lui envoie son fournisseur soit conforme avec son domaine de validité	Pour atteindre son objectif, il construit et envoie des requêtes de modification à son fournisseur	– Nb d'entrée : 1 – Nb de sortie : 0 – Nb de domaine de validité : 1 – Nb pas de modification : 0	
Boucle (AB)	– Minimiser l'écart entre la valeur de l'objectif fourni en entrée et la valeur obtenue en sortie – Rendre cohérent la valeur de sortie avec l'objectif de l'utilisateur.	Ses objectifs individuels sont de limiter la criticité due à son objectif de sortie ainsi que celle due à la différence entre son entrée et sa sortie système	Pour agir, il peut modifier la valeur du paramètre qu'il fournit au système, et envoyer des requêtes de modification au fournisseur de la sortie système	– Nb d'entrée : 1 – Nb de sortie : 1 – Nb de domaine de validité : 1 – Nb pas de modification : 1	

TAB. 4.1 – Caractéristiques des agents

4.3 L'auto-régulation du système

Dans un système complexe, un expert du domaine peut souvent comprendre l'influence d'une décision locale sur le comportement global [40]. Cependant, il lui est souvent difficile d'identifier les actions locales à effectuer pour modifier le comportement global du système [72]. Il lui est donc impossible de décrire toutes les situations que l'agent peut avoir à rencontrer et d'en extrapoler des modèles de décisions locaux basés sur son expertise. Par conséquent, la régulation des paramètres doit provenir des éléments du système et non d'un contrôle global.

4.3.1 La régulation des systèmes en cybernétique

Parce qu'un système complexe est composé d'un ensemble d'éléments en interaction, sa simulation ne peut fonctionner que si elle incorpore des mécanismes de régulation. Ces mécanismes de régulation servent à atteindre les objectifs des éléments, en coordonnant leurs actions afin de contrôler l'activité du système [41].
Lorsqu'un système est linéaire, les effets obtenus sur les sorties sont proportionnels aux modifications effectuées sur les entrées. Dans ce cas, le système respecte le principe de la conservation de l'énergie. En revanche dans un système non linéaire, la conservation de l'énergie est inapplicable, lorsque la modification d'une entrée implique des modifications multiples et non proportionnelles sur des sorties. Les phénomènes de non linéarités et la non conservation de l'énergie sont d'autant plus présents pour les systèmes composés de boucles, pour lesquels la modification d'un paramètre engendre un retour d'expérience sur la valeur du même paramètre. Par exemple, dans un écosystème, où un ensemble d'individus partagent et produisent de la ressource, si des individus viennent à modifier leur comportement, alors l'environnement et les ressources changent et donc tous les individus doivent à nouveau adapter leur comportement. Selon la nature et l'impact des comportements, l'environnent maintient ou ne maintient pas le système dans un *état d'équilibre*.
Définir des mécanismes de régulation peut permettre au système de se maintenir dans des états attendus et souhaités. Cependant, dans certaines situations, l'introduction d'une régulation trop importante peut aussi avoir des impacts négatifs en maintenant le système dans un équilibre. Plusieurs types de régulation sont possibles :

1. La régulation par *buffer* : permet de tenir compte des situations du passé dans le choix d'une action.
2. La régulation par *feedforward* : utilise des mécanismes d'anticipation sur la réaction du système.

3. La régulation par *feedback* : adapte la valeur de l'action en utilisant la réaction du système à l'action précédente.

Prenons le cas de la régulation de la température intérieure, la régulation par *buffer* fournit une indication sur la variation à appliquer en fonction des expériences précédentes. Un thermomètre extérieur permettra d'anticiper l'évolution des besoins (*feedforward*). Enfin, une information sur la dernière augmentation ou diminution de température informe sur l'inertie actuelle du système (*feedback*).

D'une manière générale, la mise en place des mécanismes de régulation aident donc un système à converger. Souvent ces mécanismes sont nécessaires et la précision accordée à leur mise en application favorise l'émergence d'un système efficace. Dans les paragraphes suivants, nous illustrons comment les agents prennent en compte le *feedback* de l'environnement en considérant des situations non coopératives.

4.3.2 Le feedback de l'environnement

Dans notre cas, la régulation consiste à trouver un jeu de paramètres qui satisfasse les contraintes émises par un utilisateur (concepteur), en utilisant les interactions entre les agents. La seule manière d'obtenir des règles de décision pour l'ensemble de ces interdépendances est de les apprendre progressivement par un processus d'essais/erreurs, et de *rétroactions (feedbacks)* positives/négatives avec l'environnement [29] [40]. Ce processus est d'autant plus important, que sans lui le système est potentiellement soumis à des oscillations ainsi qu'à des phénomènes chaotiques. Pour apprendre les interdépendances de paramètres et plus généralement pour adapter son comportement, l'agent tient compte des actions effectuées par le passé. Lorsqu'il appréhende une nouvelle situation (modification de son environnement), il compare cette situation à celles qu'il a rencontrées dans un proche passé. Grâce à cette comparaison, il est capable d'établir une relation entre les actions exécutées et leurs impacts sur son environnement. Selon la nature de cette relation, l'agent perçoit un feedback positif ou négatif.

Feedback positif. Les nouvelles actions sont dites équivalentes ou non contradictoires avec les actions précédentes, lorsque l'agent n'observe pas de nouveaux conflits en contradiction avec les actions qu'il vient de mener. Dans ce cas, il considère qu'il a contribué à améliorer l'état de son environnement. Ainsi, si une requête est équivalente à la précédente, l'agent considère que le système est satisfait. Il renforce alors son action précédente.

Feedback négatif. Inversement, si l'action demandée est contradictoire avec la précédente, alors l'action précédente a mené à une situation conflictuelle. Ainsi la rétroaction de l'environnement est négative. L'agent a alors à reconsidérer son ou ses

actions précédentes avec pour objectif de trouver un compromis, pour lequel les situations deviennent moins conflictuelles.

4.3.3 Les exigences de l'adaptation des paramètres

Comme décrit en 4.2, nos agents sont pourvus d'entrées/sorties, de domaines de validité pour chacune de leurs entrées, ainsi que de capacités de coopération et de communication. Le raisonnement coopératif (cf. 4.2.3) les incite à agir soit en fonction de leurs contraintes locales (non satisfaction de leurs domaines de validité), soit en fonction de leur perception environnementale (aider l'agent le plus critique). Cependant les principes de ce raisonnement coopératif pour assurer la convergence du système, doivent également intégrer les actions et perceptions précédentes de l'agent. Il doit donc être complété par l'identification de nouvelles situations non coopératives, qui permettent à l'agent d'adapter ses actions en fonction de *feedbacks* environnementaux, tel que décrit précédemment (cf. 4.3.2).

Ainsi pour être coopérative, la modification ou la demande de modification d'un paramètre doit suivre les règles suivantes :

Satisfaire l'agent perçu comme étant le plus en difficulté. Chaque agent reçoit des demandes en provenance de ses successeurs. En triant ces demandes par ordre de criticité, l'agent peut sélectionner la demande qui lui semble la plus critique. Un modèle étant composé de plusieurs sorties et plusieurs agents pouvant être utilisateur d'une sortie, plusieurs étapes de sélection sont nécessaires.

1. L'agent commence par sélectionner la requête de modification qu'il considère comme la plus critique pour chacune de ses sorties.
2. Ensuite, il construit les modifications correspondantes sur chacune de ses entrées en utilisant les dépendances fonctionnelles entre ses entrées/sorties (matrice jacobienne). Ainsi pour chaque entrée, l'agent dispose d'une liste de modifications.
3. Parmi la liste de modifications, il lui suffit alors de sélectionner la requête qu'il considère comme étant la plus critique pour chacune de ses entrées.

Satisfaire ses objectifs personnels. Ayant sélectionné la requête qu'il considère comme étant la plus critique, le raisonnement coopératif exige maintenant que l'action effectuée ou demandée ne provoque pas un état moins satisfaisant pour l'agent.

1. Dans le cas le plus simple, si son domaine de validité personnel est plus critique que la modification, alors il agit pour lui-même.
2. Si l'adaptation demandée par la modification implique une augmentation de sa criticité, alors il vérifie que son augmentation de criticité reste inférieure à

celle de la modification. Dans ce cas, l'agent cherche à anticiper l'impact de son action.

Adapter le paramètre en fonction des actions précédentes. Enfin pour assurer la convergence et la dynamique du système, la valeur du pas de modification permettant l'adaptation doit évoluer dynamiquement et en fonction des besoins de coopération. La progression du pas de modification de chaque entrée dépend donc du feedback environnemental perçu par l'agent. Par exemple, si un agent fait le choix d'augmenter la valeur d'une de ses entrées et que par la suite il reçoit un feedback qu'il juge positif, alors il aura tendance à poursuivre cette adaptation et à augmenter son pas de modification. À l'inverse, si une adaptation lui semble non bénéfique pour son environnement, il aura tendance à diminuer son pas de modification pour favoriser la convergence vers un consensus.

4.3.4 La modification du pas : un apprentissage progressif

Pour adapter son pas de modification, chaque agent utilise un ensemble d'actions locales, basées sur sa perception environnementale. Parce qu'il est conceptuellement distribué, l' apprentissage est intéressant, et permet au système d'être robuste, ouvert aux changements, et à la dynamique (voir partie III chapitre 8.2). Cependant pour être efficace, cet apprentissage doit être à la fois progressif et dynamique, car la perception environnementale d'un agent est parfois incomplète voire incohérente.

L'incomplétude. La perception environnementale d'un agent est limitée aux actions qui lui semblent les plus critiques à un moment donné. À cause des discontinuités, de sa vue partielle du système, l'agent est contraint d'apprendre les interdépendances par un jeu d'essais/erreurs. Ses décisions locales sont donc souvent remises en cause par l'apparition de situations non coopératives dont il ne peut anticiper l'apparition. De ce fait, chaque agent est amené à faire des choix qui ne s'avèrent pas toujours judicieux mais qui selon sa connaissance à un moment donné lui semble être coopératifs.

L'incohérence. Une action est perçue comme coopérative ou non coopérative par un agent en fonction du *feedback* environnemental qu'il perçoit. Cette perception incomplète le conduit parfois à de fausses interprétations. Par exemple, l'augmentation ou la diminution de la criticité peut ne pas être due à son action individuelle mais plutôt à celle d'autres agents ayant agi en parallèle. Parce que le raisonnement est distribué et local, les agents peuvent donc parfois être induits en erreur.

Le pas d'adaptation des agents doit donc tenir compte de ces caractéristiques, et permettre un apprentissage incrémental et dynamique. L'adaptation du pas permet à

l'agent de renforcer ses actions dans le cas de feedbacks positifs et de les limiter dans le cas de feedbacks négatifs.

Soit $\gamma^{Aq,xi}(t)$, le pas de modification de l'entrée x_i d'un agent A_q à un instant t, et $\psi^{Aq,xi}$ son facteur de renforcement.

1. Dans le cas d'un *feedback* positif, on a :

   Si $\gamma^{Aq,xi}(t) < \gamma^{Aq,xi}_{Max}$, alors

   $$\gamma^{Aq,xi}(t+1) = \gamma^{Aq,xi}(t) + \psi^{Aq,xi} \tag{4.5}$$

   Remarque : Le facteur de renforcement $\psi^{Aq,xi}$ est fortement inférieur au pas d'adaptation minimal $\gamma^{Aq,xi}_{Min}$ ($\psi^{Aq,xi} \ll \gamma^{Aq,xi}_{Min}$), afin de s'assurer que l'augmentation du pas de modification est relativement lente.

2. Dans le cas d'un *feedback* négatif :

   Si $\gamma^{Aq,xi}(t) > \gamma^{Aq,xi}_{Min}$, alors

   $$\gamma^{Aq,xi}(t+1) = \frac{\gamma^{Aq,xi}(t)}{K} \tag{4.6}$$

   Remarque : Pour éviter les oscillations, la valeur de K doit être beaucoup plus grande que 1 ($K \gg 1$). Ainsi dès l'apparition d'un conflit, le pas d'adaptation diminue fortement par division. Ceci aide l'agent à se diriger rapidement vers une valeur de consensus.

4.3.5 La mesure de la confiance

Comme décrit dans le raisonnement coopératif, les agents s'échangent des demandes de modification afin de satisfaire les contraintes en les régulant. Cette régulation tient compte des *feedbacks* environnementaux et permet de converger vers des états non contraints. Cependant, la topologie du système et le fait que les demandes soient transmises de proche en proche conduisent parfois à ce que des demandes provenant d'une même origine soient alternativement conflictuelles. Ceci provient des non linéarités du système (propagation de gradients non cohérents, présence de boucles, etc.). Dans ce cas, et d'un point de vue local, l'agent constate des demandes contradictoires provenant d'un seul et même objectif. De son propre point de vue, il doit considérer ce type de demande comme non coopérative. Pour cela, l'agent possède une mémoire des demandes de modifications récentes qu'il a reçues pour chaque agent initiateur de modification. En utilisant cette mémoire, il calcule

la confiance qu'il accorde à l'agent. Si cette confiance est inférieure à un certain seuil, les requêtes provenant de cet agent sont ignorées au profit d'autres demandes moins critiques.

Soit δt, l'intervalle de temps pendant lequel les requêtes de modification sont mémorisées. Si on note $m_j^{Aq,xi}$ une nouvelle modification j, provenant de l'entrée x_i d'un agent A_q. Si cette demande est positive, la mesure de sa confiance vaut alors :

$$Conf_{m_j^{Aq,xi}Positive} = \frac{\sum_{\delta t} m_k^{Aq,xi}Positive}{\sum_{\delta t} m_k^{Aq,xi}Positive + m_k^{Aq,xi}Negative} \tag{4.7}$$

A l'inverse si cette demande est négative, la mesure de sa confiance vaut :

$$Conf_{m_j^{Aq,xi}Negative} = \frac{\sum_{\delta t} m_k^{Aq,xi}Negative}{\sum_{\delta t} m_k^{Aq,xi}Positive + m_k^{Aq,xi}Negative} \tag{4.8}$$

4.3.6 La gestion des boucles de rétroactions

Un agent boucle de rétroactions a pour rôle la gestion d'un paramètre qui est à la fois en entrée et en sortie du système. Tel que décrit précédemment, son rôle est d'assurer la convergence de ces deux valeurs tout en respectant les objectifs du concepteur sur ce paramètre. Comme pour les autres agents sa perception environnementale reste locale.

Son double rôle est important et délicat :

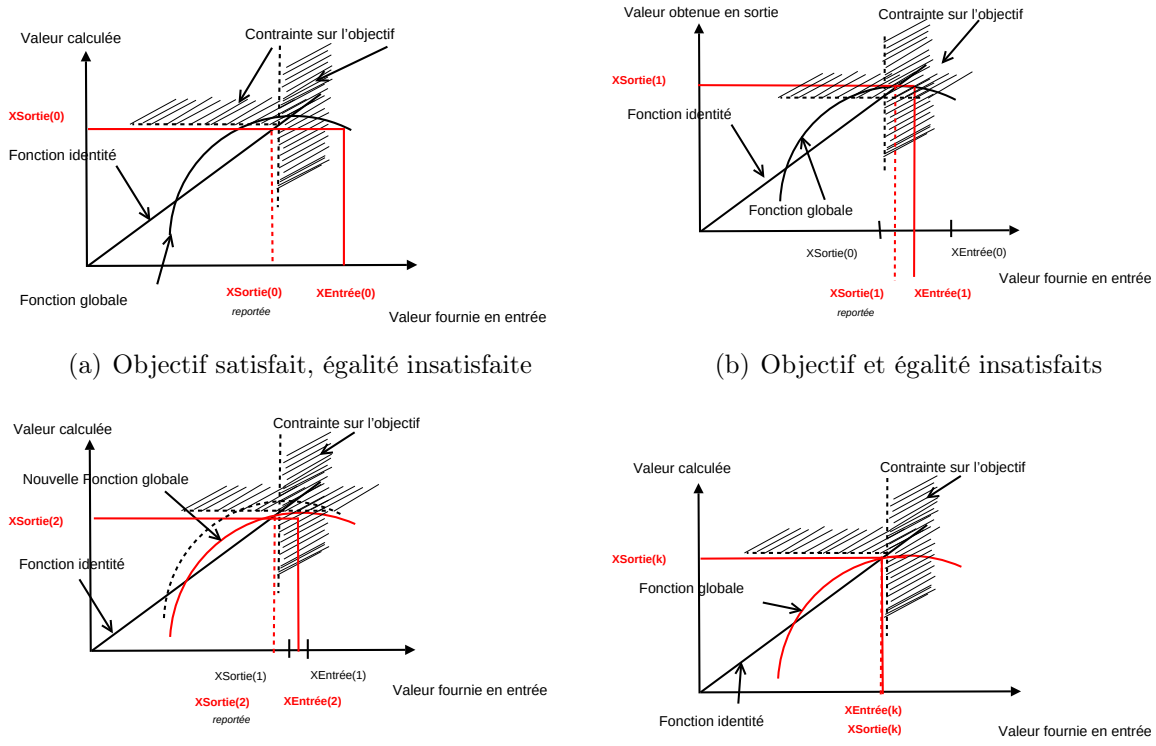
1. La convergence des deux paramètres est indispensable, car sans elle les solutions trouvées par le système sont incohérentes (physiquement invalides). Or même si le système est critique, il est important qu'il soit cohérent pour trouver des adaptations de paramètres qui aient un sens physique et qui permettent d'envisager une amélioration de l'état général du système qui soit progressive et cohérente.
2. En même temps, une prise en compte trop forte de cette contrainte d'égalité peut aussi avoir des effets négatifs, notamment lorsque la valeur obtenue en sortie ne respecte pas les contraintes de l'objectif. Faire évoluer la valeur vers l'espace des solutions admissibles est donc également important, car on veut aussi faire disparaître les situations critiques définies par les plages de validité des modèles.

La figure 4.5 illustre quelques situations que nous détaillerons par la suite et qui sont potentiellement rencontrées par un agent boucle. Elles illustrent le besoin de résoudre les deux problèmes introduits précédemment à savoir : satisfaire les contraintes du concepteur, et obtenir une égalité de valeur en entrée/sortie du paramètre bouclé.

La courbe *fonction globale* y représente la fonction qui permet à partir d'une valeur de l'entrée bouclée, de calculer la valeur de sa sortie. Cette fonction est dépendante des choix de valeurs faits sur les autres paramètres du système. Ainsi lorsque l'on change le paramétrage du système, on change la forme de cette fonction. Finalement une solution

au problème est atteignable, lorsque cette fonction globale croise la fonction identité dans le *domaine admissible* de l'objectif, tel que cela est illustré sur la figure 4.5(d). Pour que la solution soit effectivement trouvée, il faut donc que la valeur fournie en entrée soit égale à celle de l'intersection entre *les fonctions globale et identité*. Dans ce cas, la valeur fournie en entrée est bien égale à la valeur obtenue en sortie et l'ensemble des contraintes sont respectées.
La difficulté de ce problème est que la fonction globale est inconnue, il faut donc adapter les demandes et actions de l'agent boucle en utilisant uniquement sa perception environnementale locale. Voici donc quelques situations et moyens d'action utilisés par l'agent boucle pour permettre la convergence de sa valeur bouclée.

1. **Analyse de la situation** : La figure 4.5(a) représente un état initial, pour lequel la valeur *XEntrée(0)* est appliquée au système. Il en ressort la valeur *XSortie(0)* qui est différente de la valeur *XEntrée(0)*, mais à l'intérieur de l'espace admissible de l'objectif. L'agent ne connaissant pas la forme de la *fonction globale*, il ne peut pas savoir à ce moment de la résolution que la solution au problème n'est pas atteignable (puisque l'intersection entre *les fonctions globale et identité* n'est pas dans l'espace admissible de l'objectif).
   Action : Dans ce cas de figure, l'agent modifie donc uniquement sa valeur d'entrée en choisissant une valeur (*XEntrée(1)*) comprise entre la valeur précédemment introduite (*XEntrée(0)*) et celle que le système a calculée (*XSortie(0)*).
2. **Analyse de la situation** : La situation obtenue à l'étape suivante est représentée en figure 4.5(b). La valeur *XEntrée(1)* est appliquée au système. Il en ressort la valeur *XSortie(1)*. Cette fois-ci la valeur obtenue est non seulement différente de *XEntrée(1)* mais aussi à l'extérieur de l'espace admissible de l'objectif.
   Action : Dans ce cas, l'agent boucle choisit donc non seulement d'adapter son entrée (comme précédemment) mais aussi d'envoyer une demande de modification aux autres agents pour qu'ils modifient les valeurs de leurs paramètres, exactement comme le ferait un *agent objectif de sortie*. En modifiant leurs paramètres, les autres agents changent la forme de *la fonction globale* et vont ainsi participer à la réussite de la convergence du bouclage.
3. **Analyse de la situation** : À l'étape suivante (figure 4.5(c)), la forme de *la fonction globale* a changé à la suite des demandes de modifications de paramètre envoyées et réalisées par les autres agents. Cependant, une différence subsiste entre les valeurs (fournie/calculée) du paramètre de la boucle, ce qui implique les actions suivantes.
   Actions : Des modifications successives sont réalisées sur la valeur introduite dans le système, de façon à choisir à chaque fois une valeur comprise entre celle fournie au système et celle effectivement calculée (comme lors de l'étape 1).



(a) Objectif satisfait, égalité insatisfaite

(b) Objectif et égalité insatisfaits

(c) Objectif satisfait, égalité insatisfaite

(d) Objectif et égalité satisfaits

FIG. 4.5 – Les contraintes sur la valeur d'une boucle de rétroactions

4. **Analyse de la situation** : La figure 4.5(d) présente un état satisfaisant l'ensemble des contraintes. L'agent boucle ne modifie plus la valeur de son entrée et n'envoie pas non plus de demandes de modification, puisque la valeur du bouclage a convergé vers une valeur admissible.

Pour résumer, l'objectif pour obtenir la convergence d'une boucle est bien de chercher à faire converger les valeurs du paramètre (fournie et calculée). Mais pour obtenir cette convergence, on doit souvent modifier simultanément la valeur d'autres paramètres d'entrée du système afin de modifier la forme de *la fonction globale* et permettre d'obtenir un point de convergence qui soit à l'intérieur de l'espace des solutions du paramètre bouclé. Ainsi l'agent boucle :

- propage des demandes de modification en direction des entrées systèmes pour modifier la fonction globale.
- adapte la valeur qu'il fournit en entrée du système, afin de la faire progressivement converger vers la valeur effectivement calculée.

4.3.7 Synthèse des situations non coopératives

Le tableau 4.2 résume les situations non coopératives illustrées et utilisées dans notre algorithme de régulation de contraintes.

Situations non-coopératives	**Conditions**	**Actions**
L'environnement de l'agent est plus critique	Une des requêtes reçues indique qu'un agent de l'environnement est plus critique que la situation locale de l'agent	L'agent agit pour satisfaire l'environnement (sélection de la modification la plus critique)
Augmentation de la criticité locale	La modification demandée par l'environnement implique une augmentation de la criticité locale supérieure à celle de la modification.	L'agent doit diminuer la demande de modification pour ne pas devenir lui même le plus critique (adaptation du pas de modification)
Les demandes provenant de l'environnement sont changeantes	La modification demandée par l'environnement change de sens.	L'agent doit diminuer l'ampleur de ses modifications pour tendre vers un consensus, pour lequel la criticité des agents est la plus faible (adaptation du pas de modification)
Les demandes provenant d'un agent sont changeantes	La modification demandée par un agent de l'environnement change de sens.	Les modifications effectuées localement sont alors inutiles. Les demandes en provenance de cet agent doivent être ignorées (mesure de la confiance)

TAB. 4.2 – Situations non coopératives

4.3.8 L'algorithme retenu pour l'auto-régulation du système

En utilisant les agents, les mécanismes de coopération et de régulation présentés précédemment, nous déduisons les procédures et fonctions principales qui découlent de l'ensemble de ces choix. Deux procédures sont dédiées aux agents disciplinaires, qui ont à gérer à la fois des messages d'exécution *RéceptionMessageExécutionAgentDisciplinaire* et de modification *TraitementModificationAgentDisciplinaire*. La procédure *TraitementModificationsAgentObjectifEntrée* permet aux agents objectifs d'entrée de gérer les messages de modification, qu'ils reçoivent. La procédure *TraitementExécutionAgentObjectifSortie* génère des modifications en provenance des agents objectif de sortie. Enfin la procédure *RéceptionMessageModificationAgentboucle* permet aux agents boucles d'obtenir la convergence des valeurs à la fois entrées/sorties du système.

Procédure *TraitementModificationAgentDisciplinaire*
début

1. **Pour chaque** sortie j de l'agent **faire**
   (a) *modification-retenue* = RecevoirRequêtes-Et-SélectionnerPlusCritique (j) ;
2. **Pour chaque** dépendance de la sortie j avec une entrée i **faire**
   (a) construire l'objectif pour l'entrée i, correspondant à la *modification-retenue*

   fait
3. **Pour chaque** entrée i de l'agent **faire**
   (a) *objectif-retenue* = choisir l'objectif le plus critique de l'entrée i
   (b) **si** l'entrée i n'a pas de fournisseur **alors**
      i. adapter la valeur de l'entrée en accord avec l'*objectif-retenue*
   (c) **sinon**
      i. envoyer une demande de modification en accord avec l'*objectif-retenue* à l'agent fournisseur

      fin si

   fait

fin

Fig. 4.6 – Procédure de traitement des messages de modification d'un agent disciplinaire

La procédure *TraitementModificationAgentDisciplinaire*

L'algorithme de la figure 4.6 résume la phase de gestion des modifications par un agent disciplinaire. En faisant appel à la fonction RecevoirRequêtes-Et-SélectionnerPlusCritique

(fig 4.7), l'agent reçoit les demandes de modification provenant des consommateurs de chacune de ses sorties (fig 4.6, étape (1)) et sélectionne pour chaque sortie la demande la plus critique (fig 4.7, (2)). En utilisant les relations de dépendances entre ses entrées/sorties, il construit la requête associée au niveau de chaque entrée (fig 4.6, (2)). Enfin, il sélectionne la demande de modification retenue pour chaque entrée (fig 4.6, (3a)) et la transmet au fournisseur (fig 4.6, (3c)) ou l'effectue (fig 4.6, (3b)). Lorsque les demandes de modifications disparaissent, tous les agents présents dans son environnement sont satisfaits. Le système a alors convergé vers un consensus, pour lequel toutes les contraintes sont satisfaites.

Fonction *RecevoirRequêtes-Et-SélectionnerPlusCritique (Sortie j)*
début

1. **tant que** toutes les requêtes de la sortie j n'ont pas été reçues **faire**
   (a) recevoir les requêtes pour la sortie j provenant des k consommateurs
   fin tant que
2. *modification-retenue* = choisir la requête la plus critique pour la sortie j parmi les k requêtes reçues

fin

Fig. 4.7 – Fonction de réception et de sélection des messages de modification d'une sortie d'un agent

La procédure *RéceptionMessageExécutionAgentDisciplinaire*

L'algorithme 4.8 détaille le traitement des messages d'exécution par un agent disciplinaire. Cette phase est simple, l'agent reçoit l'ensemble de ses messages d'exécution, puis exécute son modèle et enfin envoie les nouvelles valeurs de ses sorties à ses agents utilisateurs.

La procédure *TraitementModificationsAgentObjectifEntrée*

Le rôle d'un agent objectif d'entrée est de choisir la valeur du paramètre, qui satisfasse au mieux les demandes de ses successeurs. Cet agent ne possède aucun fournisseur de paramètre et par conséquent son raisonnement s'organise de la manière suivante :

- recevoir les demandes de modification provenant de ses consommateurs ;
- modifier la valeur du paramètre en accord avec les demandes ;
- informer ses consommateurs de la nouvelle valeur choisie.

Le détail de cet algorithme est décrit en figure 4.9.

Procédure *RéceptionMessageExécutionAgentDisciplinaire*
début

1. **tant que** tous les messages d'exécution n'ont pas été reçu pour toutes les entrées **faire**
   (a) recevoir les messages d'exécution provenant des fournisseurs
   fin tant que
2. exécuter le modèle
3. envoyer les valeurs des sorties aux consommateurs

fin

Fig. 4.8 – Procédure de traitement des messages d'exécution d'un agent disciplinaire

Procédure *TraitementModificationsAgentObjectifEntrée*
début

1. *objectif-retenu* = RecevoirRequêtes-Et-SélectionnerPlusCritique (sortieAgent) ;
2. adapterPasModification(*objectif-retenu*) ;
3. modifier la valeur de l'entrée
4. envoyer la nouvelle valeur aux agents consommateurs sous la forme d'un message d'exécution

fin

Fig. 4.9 – Procédure de traitement des messages de modification d'un agent objectif d'entrée

La procédure *TraitementExécutionAgentObjectifSortie*

À l'inverse les agents objectifs de sortie ne possèdent aucun consommateur de paramètre et leur rôle est de générer des demandes de modification, qui permettent au système de tenir compte des performances souhaitées par le concepteur. À chaque itération, leur raisonnement local suit donc les étapes décrites en figure 4.10.

Procédure *TraitementExécutionAgentObjectifSortie*
début

1. recevoir un message d'exécution
2. construire et envoyer un message de modification en accord avec le domaine de validité du paramètre

fin

Fig. 4.10 – Procédure de traitement des messages d'exécution d'un agent objectif de sortie

La procédure *RéceptionMessageModificationAgentBoucle*

Dans le sens direct (exécution), un agent boucle reçoit la nouvelle valeur calculée par le système. Il agit alors comme un agent objectif de sortie et réalise donc la procédure *TraitementExécutionAgentObjectifSortie*.

Dans le sens des demandes de modification, il reçoit les demandes de modification provenant des autres parties du système comme le ferait un agent objectif d'entrée (*TraitementModificationsAgentObjectifEntrée*).

La différence est qu'en plus de la comparaison de la criticité des modifications reçues avec celle de son objectif, il a aussi à comparer celle due à la différence de valeur obtenue entre l'entrée et la sortie du système. Il choisit alors la demande la plus critique et modifie sa valeur.

Procédure *TraitementModificationsAgentBoucle*
début

1. *objectif-modifications* = RecevoirRequêtes-Et-SélectionnerPlusCritique (sortieAgent) ;
2. *objectif-différence* = CriticitéDeLaDifference (valeurEntrée, valeurSortie)
3. *objectif-retenu* = SélectionnerPlusCritique (*objectif-modifications*, *objectif-différence*)
4. adapterPasModification(*objectif-retenu*) ;
5. modifier la valeur de l'entrée
6. envoyer la nouvelle valeur aux agents consommateurs sous la forme d'un message d'exécution

fin

Fig. 4.11 – Procédure de traitement des messages de modification d'un agent boucle

4.3.9 Déroulement de quelques séquences de raisonnement d'un agent disciplinaire

Les figures 4.12, 4.13, 4.14 et 4.15 illustrent quelques itérations du raisonnement d'un agent disciplinaire.

Lors de la première étape (4.12) :

1. L'agent reçoit des demandes de modification en provenance des sorties (P3 et P4).
2. Il construit les demandes de modification correspondantes sur ses entrées (P1 et P2). On notera que le paramètre P2 n'a pas de dépendance avec la sortie P4, il ne possède donc qu'une demande de modification provenant de la sortie P3.

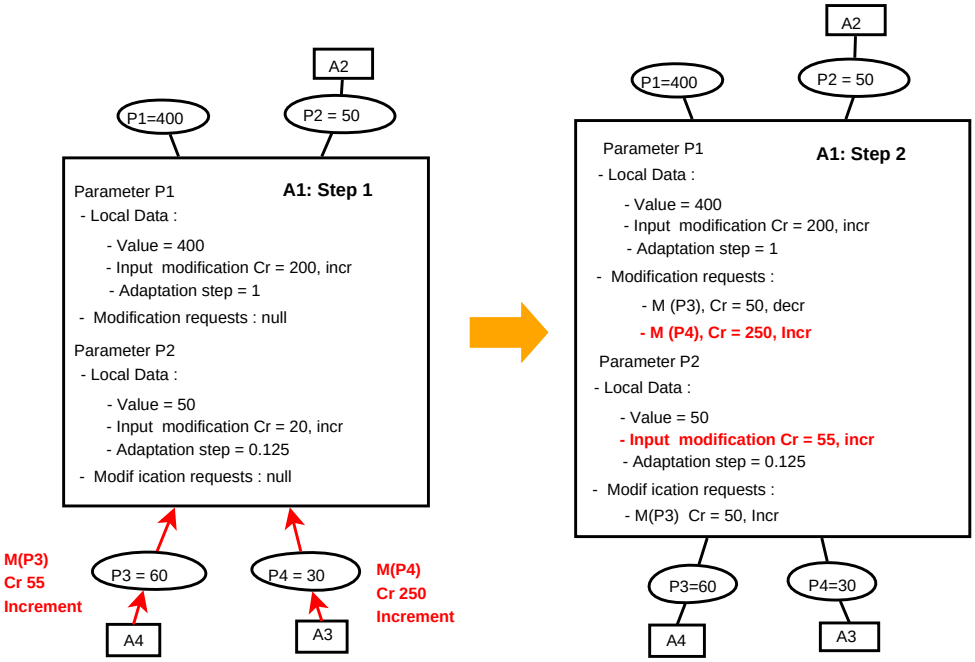


FIG. 4.12 – Déroulement de l'algorithme (étape 1 et 2).

3. Une demande de modification est choisie pour chaque entrée, en comparant les criticités des modifications reçues avec les criticités locales. Ainsi, pour le paramètre P1, la demande de modification retenue est celle provenant de la sortie P4 puisque sa criticité est supérieure à celle du paramètre P1 (200). De même pour le paramètre P2, la modification retenue est celle du paramètre P3.

Lors de la seconde étape 4.13 :

1. L'agent agit en utilisant les modifications sélectionnées, le paramètre P1 n'a aucun fournisseur. La valeur est donc directement modifiée en tenant compte de la direction demandée par la modification. Concernant le paramètre P2, la demande de modification sélectionnée est relayée à l'agent A2 qui fournit le paramètre.
2. L'agent ayant envoyé une demande de modification, il reçoit une nouvelle valeur d'exécution de l'agent A2 fournissant le paramètre P2. En tenant compte de cette nouvelle valeur, il exécute son modèle et obtient de nouvelles valeurs pour ses sorties P3 (45) et P4 (45), qu'il transmet aux agents consommateurs A4 et A3.

Lors de la troisième étape 4.14 :

1. L'agent reçoit des nouvelles demandes de modification provenant des agents consommant ses sorties.
2. Comme lors de l'étape 4.12, les demandes sont triées et sélectionnées. Durant cette troisième phase l'agent compare les nouvelles demandes de modification qu'il sélec-

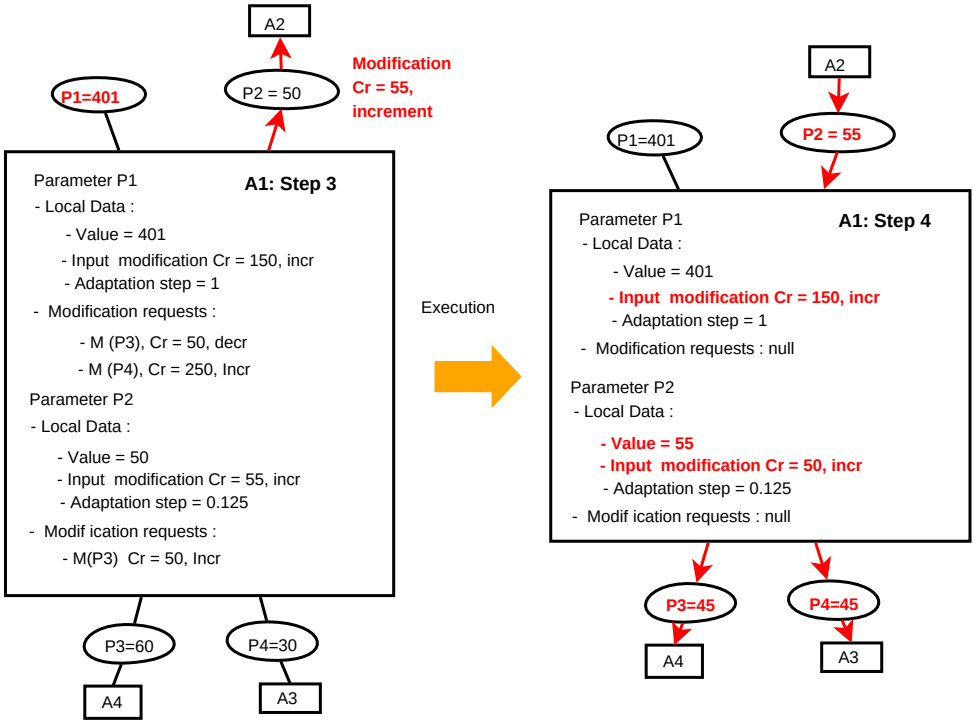


FIG. 4.13 – Déroulement de l'algorithme (étape 3 et 4).

tionne avec ses actions précédentes pour adapter son pas de modification. Concernant P1, la demande de modification la plus critique provient maintenant de P3, qui demande un changement de la direction de la modification. Le pas d'adaptation de P1 doit donc être diminué pour converger vers une valeur de consensus. À l'inverse, la demande de modification de P2 est dans la même direction, la valeur de la modification et donc celle du pas sont alors renforcées.

Finalement, la quatrième étape 4.14 est la même que la seconde. L'agent effectue la modification retenue pour P1 et transmet celle choisie pour P2 à l'agent A2 4.15.

4.4 Synthèse sur la régulation

La régulation du système est basée sur le raisonnement coopératif et sur le *feedback* environnemental. Ainsi chaque agent utilise un ensemble d'actions locales, basées sur sa perception environnementale. L'essentiel de la régulation est effectuée grâce à l'adaptation du pas de modification, qui permet au système et aux agents d'apprendre progressivement les interdépendances fonctionnelles et de converger vers des états stationnaires. Parce qu'il est conceptuellement distribué, cet apprentissage permet au système de s'auto-réguler en adaptant constamment son comportement aux changements environnementaux. Ainsi la régulation des contraintes permet à l'ensemble des agents de converger vers un consensus,

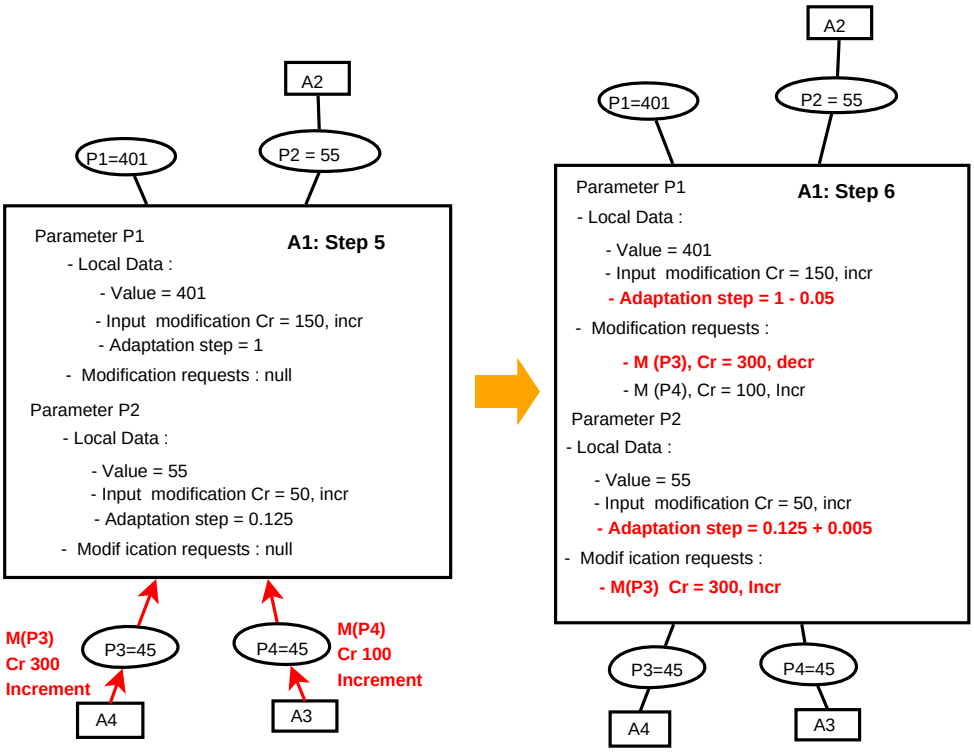


FIG. 4.14 – Déroulement de l'algorithme (étapes 5 et 6).

qui correspond à un état du système pour lequel l'ensemble des contraintes sont satisfaites. Cependant, elle ne permet donc pas de répondre à l'ensemble des besoins fonctionnels de la conception préliminaire (section 1.2.5), puisqu'elle ne fournit pas un panel de solutions équivalentes et n'optimise pas certaines des contraintes.

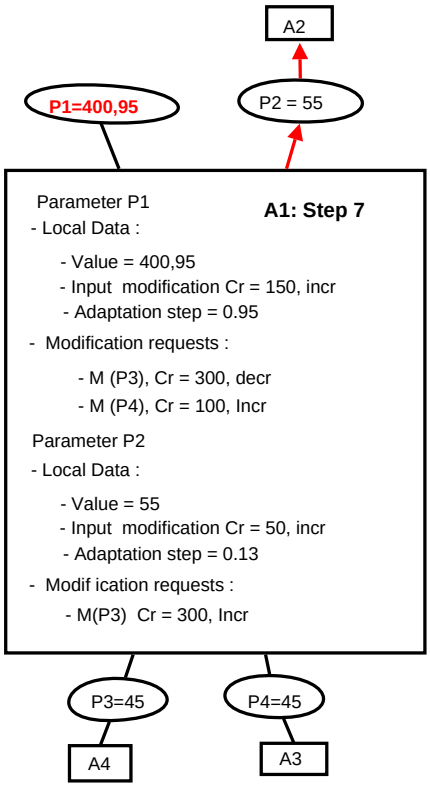


FIG. 4.15 – Déroulement de l'algorithme (étape 7).

En mathématiques, on ne comprend pas les choses on s'y habitue.

John von Neumann

5

L'optimisation des solutions

Sommaire

Comme nous l'avons vu, la régulation offre la possibilité de satisfaire un ensemble de contraintes, sans favoriser la recherche d'une solution particulière. Cette solution émerge d'une succession de négociations entre les agents du système. Lors de ce processus, les agents tirent profit des degrés de libertés qui leurs sont conférés et de leur capacité à s'adapter dynamiquement aux changements environnementaux. Mais souvent l'optimisation de problèmes multi-objectifs nécessite la mise à disposition d'un ensemble de solutions

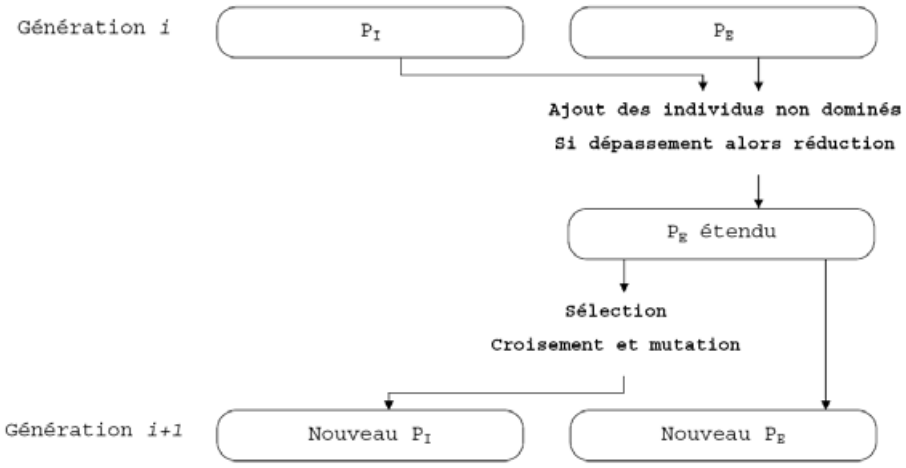


FIG. 5.1 – Principales étapes de l'approche PESA

à interpréter et à argumenter. Cette optimisation consiste à proposer des solutions sous la forme de fronts de Pareto, tel qu'introduit en section 2.1.2.
Dans cette partie nous proposons un modèle d'optimisation permettant de tirer partie de l'auto-organisation de notre système. Les notions de "décisions *a priori, a posteriori* et interactives" ainsi que d'optimisation Pareto ou non-Pareto ayant été introduites précédemment, nous commencerons par illustrer quelques algorithmes couramment utilisés.

5.1 Quelques méthodes d'optimisation multi-objectifs

Dans cette partie nous présentons des approches *a posteriori* à base de recherche Pareto (PESA [20]) et non Pareto (VECA [70]) ainsi qu'une approche interactive (iMOODS [74]).

5.1.1 Les approches Pareto

Bien souvent les méta-heuristiques qui utilisent une approche de sélection des individus non-Pareto ou Pareto, sont basées sur une mémoire des meilleurs individus. Dans les approches Pareto, cette mémoire est composée des solutions non-dominées et parfois partiellement dominées. La stratégie de ces méthodes consiste à améliorer leur population d'individus Pareto, en y ajoutant progressivement de nouvelles solutions non dominées. L'ajout de ces nouveaux individus provoque dans la plupart des cas la suppression d'autres afin d'éviter un dépassement mémoire.

Ce type d'algorithmes a inspiré la majorité des approches élitistes, telles que NSGA II [23] (*Non-dominated Sorting Genetic Algorithm-II*) et PESA [20] (*Pareto Envelope-based Selection Algorithm*). Prenons l'exemple de PESA, son algorithme est illustré en figure

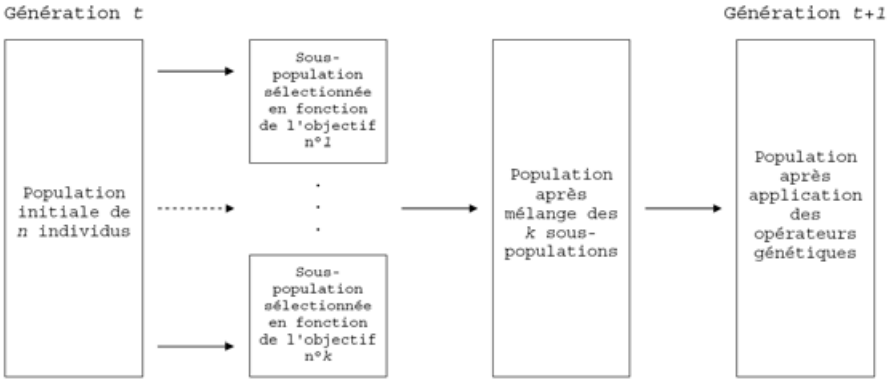


FIG. 5.2 – Principes de l'approche VEGA

5.1 et peut se résumer de la façon suivante :

1. Les individus de la population sont comparés avec ceux de l'archive contenant les individus non dominés. Si un individu de la nouvelle population est non dominé, il est ajouté à l'archive.
2. L'archive est ensuite réduite en supprimant les individus dominés et en appliquant un algorithme de clustering[4], qui assure une bonne répartition des individus sur le front.
3. Enfin une nouvelle population est calculée à partir de la population des individus non-dominés.

Ce genre de méthode est efficace mais dépendante de son paramétrage (taille de la population, taux de mutation et crossovers). Elle est donc contrainte par le nombre d'objectifs à optimiser simultanément.

5.1.2 Les approches non-agrégées et non-Pareto

En général, ces méthodes traitent séparément chacun des objectifs. C'est le cas de la méthode VEGA [70] (*Vector Evaluated Genetic Algorithm*) qui propose de créer des sous-populations, dont les meilleurs individus sont spécialisés pour l'optimisation d'un objectif particulier. Ainsi chaque sous-population possède une méthode d'évaluation adaptée à un seul objectif et converge vers des solutions spécialisées. Un second processus est utilisé afin de mélanger les solutions obtenues par les sous-populations, tel qu'illustré en figure 5.2. Ce processus permet d'obtenir des solutions de compromis.

Ce type d'approche a montré son efficacité à trouver des solutions non-dominées [10], mais la qualité des solutions de compromis est dépendante du degré d'interdépendance

[4]On calcule une distance entre les individus et on supprime une partie des individus, pour lesquels les distances avec d'autres sont les plus faibles.

entre les objectifs.

5.1.3 Les approches interactives

Les approches interactives ont l'avantage d'être co-constructives. En utilisant les connaissances et préférences du concepteur, elles limitent la recherche aux parties qui l'intéressent le plus et réduisent ainsi la dimension de l'espace de recherche. Dans iMOODS [74] (*interactive Multi-Objective Optimization Design Strategy*), la démarche proposée est la suivante :

1. génération d'un premier point de Pareto, en utilisant une pondération de critères ainsi qu'une méthode de recherche locale, type gradient ;
2. construction d'une analyse de sensibilité autour de la solution trouvée pour aider le concepteur à en comprendre les caractéristiques ;
3. invocation d'une stratégie interactive de décision : le concepteur choisit la direction de la recherche parmi les points promu par l'analyse de sensibilité. Enfin une nouvelle recherche est réalisée en utilisant ces nouveaux critères.

La méthodologie NIMBUS (*Nondifferentiable Interactive Multiobjective BUndle-based optimization System*) proposée dans [50] est similaire. Néanmoins cette approche propose de ne pas favoriser une agrégation initiale de paramètres et de rechercher plusieurs alternatives dès l'initialisation. Le concepteur a ensuite la possibilité de choisir parmi des alternatives celle qui lui semble la plus adaptée. Le choix d'une alternative entraîne alors une nouvelle recherche autour de ce point.

5.1.4 Bilan

Différentes approches ont été proposées pour faire de la recherche dans des espaces multi-objectifs. Chacune de ces méthodes a ses avantages et ses inconvénients. Ainsi, lorsque l'espace de recherche est important, on préférera souvent des approches interactives qui sont moins coûteuses et plus adaptées aux besoins du concepteur. À l'inverse pour des problèmes aux interdépendances fortes et au nombre limité d'objectifs, on pourra préférer des méthodes Pareto qui offriront une vue complète des interdépendances, ce qui favorise une bonne prise de décision.

Au regard de ces caractéristiques, nous proposons dans les sections suivantes :

1. une approche permettant d'optimiser nos solutions. Ceci nous permet de réaliser le passage d'un problème de satisfaction de contraintes à un problème d'optimisation ;
2. une méthode permettant le parcours de fronts de Pareto.

Nous commencerons par évoquer la démarche qui nous a amenés à faire nos choix d'architecture.

5.2 Les enjeux et les impacts sur la modélisation

Dans notre approche d'auto-régulation, les criticités correspondent à des valeurs de contraintes non satisfaites. La non satisfaction d'une contrainte est un critère prioritaire à l'optimisation d'un objectif, puisque l'on cherche avant tout une solution au problème. Lorsqu'un état satisfaisant les contraintes est atteint, on cherche d'autres qui minimisent ou maximisent certains objectifs, et dans ce cas réaliser une optimisation du problème.
Pour notre système, le passage à l'optimisation doit respecter deux critères :

Ne pas interférer avec la régulation. Ce premier critère paraît évident, car l'optimisation peut être vue comme un processus venant après celui de la satisfaction de contraintes. Cependant même en agissant dans un second temps, il faut vérifier que les mécanismes de coopération mis en place n'interfèrent pas avec la satisfaction de contraintes.

Pour optimiser certains critères, une solution envisagée a été d'ajouter d'autres messages demandant la diminution et/ou l'augmentation de ces paramètres et de tenir compte de ces messages uniquement, lorsque l'environnement d'un agent n'émet plus de situations critiques (criticité) liées à la non satisfaction de contraintes. Traiter le problème de cette manière revient donc à ajouter de la sémantique dans les échanges de messages, puisque certaines demandes concernent la prise en compte de non-satisfaction de contraintes et d'autres l'optimisation de certains paramètres. Implémentée de cette manière, cette approche est délicate à traiter, car le mélange de sémantique peut introduire des discontinuités dans le raisonnement des agents et donc du désordre. Par exemple, lorsque l'environnement d'un agent est satisfait et que celui-ci accepte de prendre en compte une demande d'optimisation, les choix qu'il va faire pour satisfaire cette demande vont potentiellement faire réapparaître de nouvelles situations critiques (non satisfaction de contraintes), qui automatiquement vont reprendre le dessus dans le raisonnement de l'agent.

Ainsi à cause des non-linéarités des systèmes considérés, les demandes d'optimisation ne sont jamais durablement prises en compte, ce qui fait échouer la recherche de solutions optimisées. Naturellement, on pourrait chercher des solutions à ce problème en améliorant les mémoires et les perceptions de l'agent. Mais ces améliorations ne sont pas évidentes à formuler, car elles doivent éviter les discontinuités engendrées par l'alternance de demandes de modification concernant des problèmes d'optimisation et de satisfaction de contraintes.

Éviter la pondération des objectifs. Le second critère à respecter est la non pondération des objectifs. Dès lors que l'on veut proposer un panel de solutions et non une solution optimale issue d'un agrégat de préférences, il est nécessaire de pouvoir considérer les solutions obtenues pour l'optimisation de chaque objectif. En optimisant les critères à l'aide de priorités, le système va naturellement converger vers un consensus satisfaisant plusieurs paramètres en même temps et ceci en adéquation avec les priorités définies. Mais dans ce cas, le système se déplace vers une zone particulière du front de Pareto. Or l'objectif n'est pas d'obtenir une solution, mais bien un ensemble de solutions à la manière d'un front global. Pour répondre à cette exigence, l'optimisation simultanée de plusieurs objectifs doit donc être évolutive et permettre le parcours des solutions du front de Pareto.

5.3 L'optimisation par la modification des contraintes

Pour traiter ces deux exigences et éviter les interférences avec la régulation ainsi que la pondération de critères, nous avons choisi d'implémenter un niveau de supervision. Ce niveau de contrôle n'intervient pas directement dans le raisonnement des agents mais il modifie certains paramètres de l'environnement afin de favoriser l'émergence de comportements préférés. Cette approche est partagée par d'autres comme [68], pour qui les systèmes auto-organisés peuvent requérir l'utilisation d'un niveau de contrôle supérieur, dès lors que l'on cherche à obtenir un comportement émergent qui satisfasse des propriétés préférées.
De cette manière, nous utilisons les capacités d'adaptation des agents pour optimiser les solutions, sans avoir à mélanger différents types de modifications.
L'optimisation d'un objectif est alors réalisée de la manière suivante :

1. Lorsque le système atteint un état pour lequel l'ensemble des contraintes sont satisfaites, la plage de validité de l'objectif à optimiser est modifiée afin de renforcer sa contrainte.
2. Ensuite si le système est capable de trouver par régulation une nouvelle solution au problème de satisfaction de contraintes, alors la solution est meilleure que la précédente par rapport à l'objectif considéré et on peut alors recommencer l'étape 1. À l'inverse lorsque le système est incapable d'atteindre ce nouvel état (le délai de résolution est dépassé), on peut considérer que la modification appliquée à l'objectif, a sur-contraint le système et que la valeur précédente de l'objectif était optimale.

Selon cette approche, optimiser certains critères revient donc à modifier leurs domaines de validité et à laisser le système chercher un nouvel état non contraint. Mais tel que conçu, seul un observateur extérieur peut savoir si les agents ont trouvé un état satisfaisant les

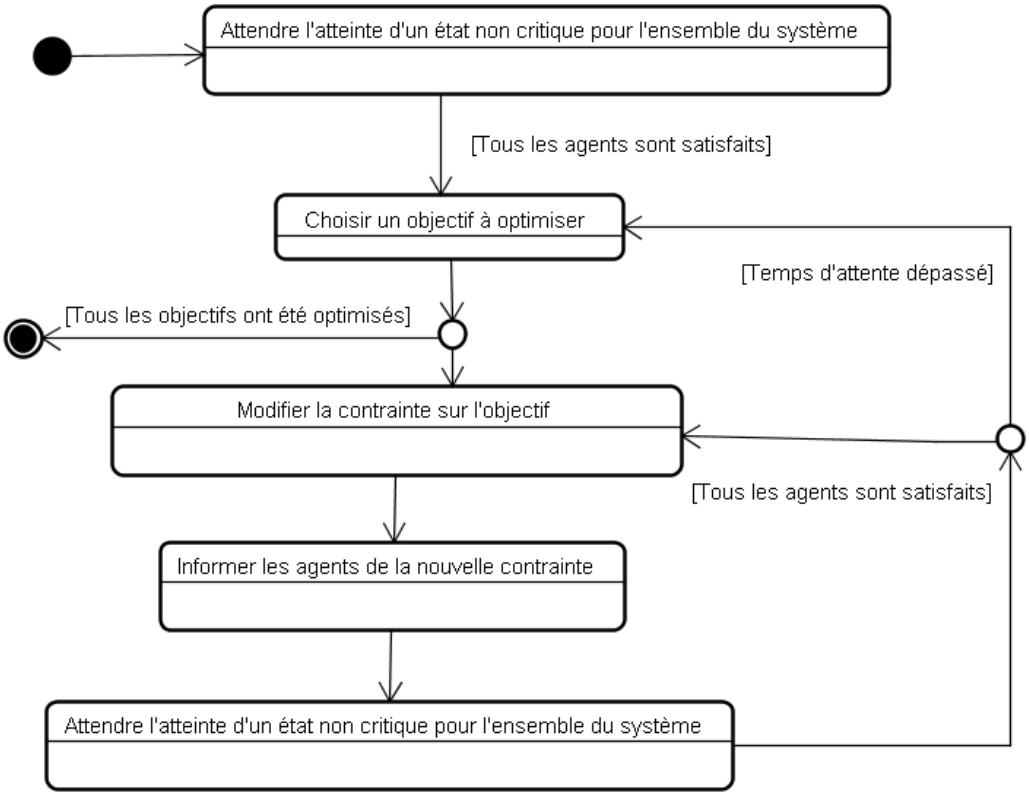


FIG. 5.3 – Optimisation des objectifs par l'agent architecte

contraintes. Ainsi il est également le seul à savoir si le système a convergé et à pouvoir évaluer l'effet de la modification. Ceci justifie le besoin d'un niveau de contrôle supérieur et donc la présence d'un agent architecte, responsable :

- de la collecte des informations provenant des agents ;
- de l'analyse du comportement global du système ;
- et finalement de l'application des modifications nécessaires au pilotage du système et à l'amélioration de certains objectifs.

Ces modifications entraînent ainsi l'émergence d'un nouveau comportement des agents. En régulant les nouvelles contraintes, les agents favorisent l'optimisation de certains objectifs. Mais finalement, l'agent architecte ne fait que tester des formulations de problème différentes et il n'agit ni sur la régulation des contraintes ni sur la résolution du problème.

Le diagramme d'états de la figure 5.3 représente le mode d'action de l'agent architecte pour optimiser un objectif.

Méthodes	**Gradient**	**AG**	**adaptive AG**	**co-evol AG**	**MASCODE**
Meilleure valeur	-30373.9	-30183.5	-30903.8	-31020.8	-30750
Moyenne	N/A	N/A	-30442.1	-30984.2	N/A

TAB. 5.1 – Résultats d'optimisation appliqués au problème d'Himmelblau [18]

Test et comparaison des résultats appliqués à la fonction Himmelblau

Pour valider la capacité de cette approche à trouver des solutions à un problème d'optimisation mono-objectif, nous avons comparé ses résultats sur un cas tests proposé par Himmelblau en 1972 [18] et pour lequel de nombreuses méthodes ont été utilisées. L'objectif ici est de vérifier si notre approche est capable de converger vers de bonnes solutions. Ce problème est composé d'un ensemble de quatre fonctions non-linéaires partageant cinq degrés de libertés. Parmi ces fonctions, trois d'entre-elles possèdent des contraintes, et la dernière un objectif, à savoir minimiser sa valeur de sortie. Il s'agit donc d'un problème mono-objectif, qui a été évalué par de nombreuses méthodes. Nous avons comparé les résultats fournis par notre approche, en associant un agent à chacune des fonctions et en minimisant la valeur de l'une d'entre elle par la modification progressive de son domaine de validité. Nous avons obtenu les résultats figurant dans le tableau 5.1.

Ces résultats montrent la capacité de MASCODE à trouver de bonnes solutions au problème d'Himmelblau, puisque les solutions obtenues sont meilleures que celles de certains algorithmes de gradient ou génétiques, et moins bonnes que celles d'algorithmes génétiques plus élaborés, dont le paramétrage s'adapte dynamiquement ou fait co-évoluer plusieurs populations. Mais ces algorithmes sont souvent moins génériques que MASCODE, puisqu'ils nécessitent un ajustement de leurs paramétrages en fonction de la nature du problème.

5.4 La stratégie pour le parcours du front de Pareto

L'approche Pareto permet de comparer des solutions multi-objectifs (cf. section 2.1.2). Dans la section précédente, nous avons proposé un modèle d'optimisation des paramètres l'un après l'autre.
Cette solution permet d'atteindre les meilleures solutions pour chacun d'entre eux. En revanche, elle ne permet pas de garantir que les optima trouvés soient bien des solutions non dominées pour l'ensemble des objectifs. La partie **(a)** de la figure 5.4 illustre ce propos sur un problème à deux dimensions, puisque la solution trouvée est optimale pour l'objectif 2 (sa valeur minimale est atteinte), et pourtant elle n'est pas sur le front de Pareto, puisqu'il existe de meilleures valeurs pour l'objectif 1.

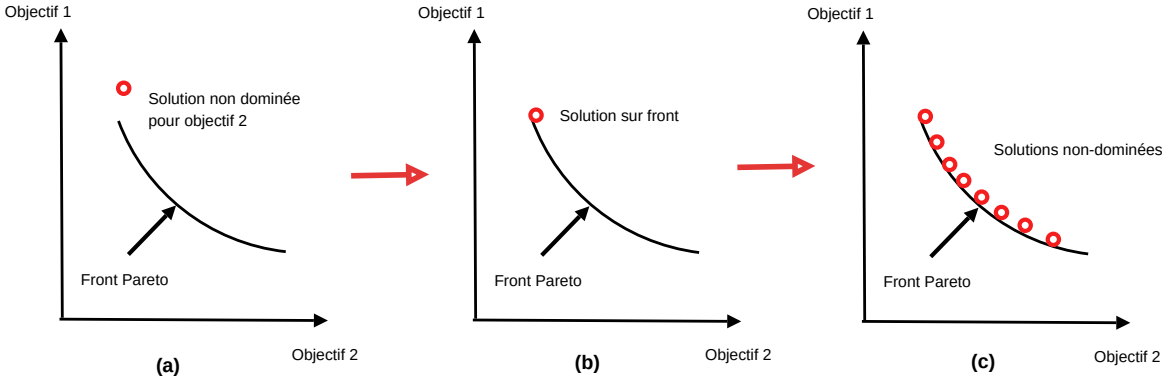


FIG. 5.4 – Caractéristiques d'une solution non-dominée en deux dimensions

Pour obtenir une solution du front, il faut donc chercher systématiquement à optimiser chacun des sous-objectifs à la manière des approches lexicographiques [19] pour être certain que la solution obtenue ne puisse être modifiée par l'amélioration d'un des sous-objectifs. Prenons l'exemple de la figure 5.4 après avoir optimisé l'objectif 2 (partie (a)), on optimise l'objectif 1, et on obtient ainsi la solution non-dominée présentée en partie centrale (b) de la figure. Mais avec cette stratégie de parcours, on ne peut obtenir que des points solutions extrêmes du front. Ainsi une autre stratégie est nécessaire si l'on veut proposer des solutions pour lesquelles les valeurs des objectifs sont des compromis, et pour trouver des solutions sur la partie centrale du front.

Pour répondre à ce problème, on adopte une nouvelle stratégie. Elle consiste à appliquer une succession de modifications afin de parcourir des solutions intermédiaires. On procède de la manière suivante :

- On dégrade le premier objectif de la solution obtenue précédemment pour se dégager du front. Cette première étape est illustrée par les deux premières parties (a et b) de la figure 5.5.
- Ensuite on optimise à nouveau chacun des sous-objectifs en changeant l'ordre dans lequel on les optimise. Cette seconde étape nous permet d'obtenir un nouveau point du front, tel qu'illustré dans la troisième partie (c) de la figure 5.5. En changeant aléatoirement l'ordre de sélection des sous-objectifs, on assure une certaine variété des solutions trouvées, puisque deux ordonnancements différents ne conduisent pas forcément au même compromis.

En procédant de cette manière on se déplace dans l'espace des solutions, on mémorise un ensemble de solutions non-dominées, et on obtient des consensus entre les objectifs. Cette stratégie est décrite graphiquement par la figure 5.7, ainsi que par l'algorithme commenté de la figure 5.6.

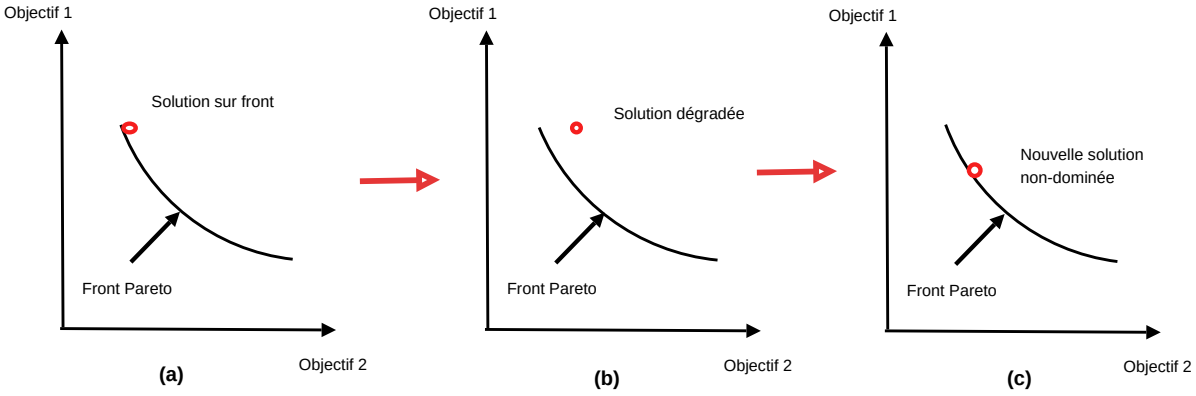


FIG. 5.5 – Stratégie de parcours du front en deux dimensions

5.4.1 Synthèse : l'agent architecte

Son rôle

Étant donné la stratégie de résolution que nous avons choisie, le rôle de l'agent architecte est assez simple. Il se résume finalement à appliquer des modifications aux agents responsables des objectifs à optimiser, à observer leurs réactions à ces changements et à mémoriser les nouvelles solutions obtenues.

L'efficacité du parcours des solutions réside dans la manière de sélectionner les paramètres à modifier. Cette stratégie est importante, car c'est elle qui permet au système de trouver des solutions Pareto-optimales et de parcourir le front efficacement.

La stratégie que nous avons définie est assez générique. Mais l'idéal est de définir une stratégie en accord avec la nature du problème, et d'utiliser les préférences de l'utilisateur tel que nous le discuterons en section 5.7.

Ses connaissances

Concrètement, les choix de cette stratégie supposent que l'agent chargé de l'optimisation, connaisse les objectifs à optimiser, qu'il ait une idée de l'évolution du système et qu'il puisse appliquer des modifications sur les intervalles objectifs de certains paramètres. Il a donc une mémoire des objectifs optimisés, une mémoire des solutions de Pareto que le système a déjà parcourues, ainsi qu'une mémoire des modifications environnementales, qu'il a effectuées.

Ses capacités

Les capacités de cet agent sont assez simples :

Procédure *Parcours du front de Pareto*
début
// On choisit un premier objectif parmi la liste.

1. **tant que** *premier-objectif* = Choisir-un-premier-objectif(*liste-objectifs*)
   // On optimise le premier objectif tant que cela est possible.
   // But : obtenir une solution non-dominée pour cet objectif.
   (a) Initialisation : *réussite-optimisation* = **vrai**
   (b) **tant que** *réussite-optimisation* = **vrai**
      // L'optimisation d'un objectif consiste à le contraindre,
      // en réduisant la largeur de son intervalle objectif.
      i. *réussite-optimisation* = Optimiser(*premier-objectif*)
   (c) **fin tant que**
   (d) Initialisation : *nombre-de-dégradation* = 0
   (e) **tant que** le *nombre-de-dégradation* < *nombre-dégradation-MAX*
      // On dégrade la solution obtenue pour le premier objectif.
      // Remarque : la dégradation est l'opération inverse à l'optimisation.
      // Elle consiste à ré-augmenter la taille de l'intervalle objectif du paramètre
      // But : se dégager du front pour trouver de nouvelles solutions.
      i. Dégradation (*premier-objectif*)
      ii. *nombre-de-dégradation*++
         // Tant qu'il reste des sous-objectifs à optimiser,
         // on en choisit un et on l'optimise.
         // But : obtenir une solution de Pareto.
      iii. **tant que** *sous-objectif* = Choix-aléatoire-sous-objectif(*liste-objectifs*) ;
         A. **tant que** Optimiser(*sous-objectif*) = vrai
         B. **fin tant que**
      iv. **fin tant que**
   (f) **fin tant que**
2. **fin tant que**

fin

Fig. 5.6 – Algorithme de parcours du front de Pareto

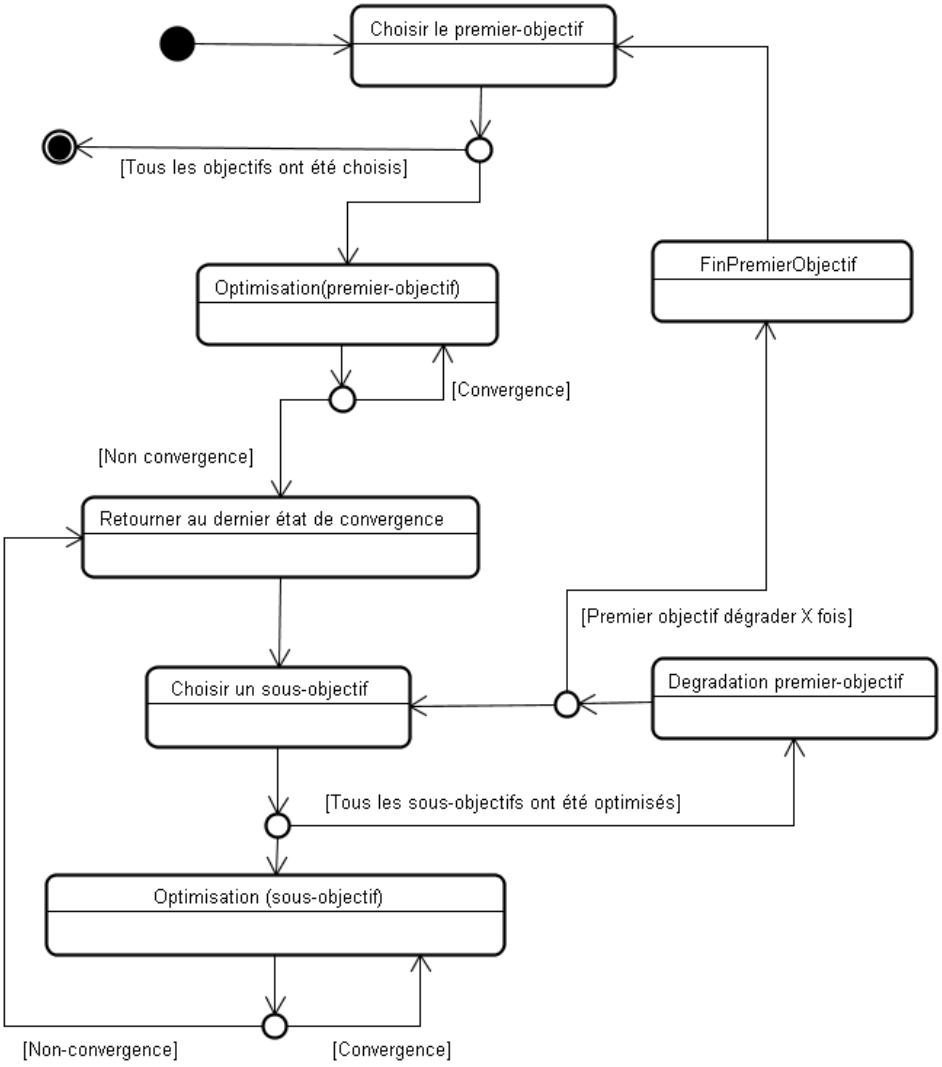


FIG. 5.7 – Parcours du front de Pareto par l'agent architecte

1. il observe le système, en interrogeant les agents sur leur état à des intervalles de temps réguliers ;
2. il optimise un objectif en envoyant des messages à l'agent concerné à l'aide de la procédure présentée en figure 5.3 ;
3. il parcourt et mémorise des solutions non-dominées en utilisant la procédure présentée en figure 5.7.

5.4.2 Étude d'un cas théorique

Pour comparer et tester notre stratégie, nous avons utilisé une fonction de référence proposée par [22]. Ce problème a la structure suivante : l'objectif est de minimiser, les fonctions f_1 et f_2 telles que :

$$f_1(x) = f(x_1) \tag{5.1}$$

$$f_2(x) = g(x_2) \star h(f, g) \tag{5.2}$$

$$g(x_2) = 1 + 10x_2 \tag{5.3}$$

$$h(f, g) = 1 - \sqrt[\alpha]{\frac{f}{g}} - \frac{f}{g} sin(2\pi q f) \tag{5.4}$$

$$\alpha = 2 \texttt{ et } q = 5 \tag{5.5}$$

L'intérêt de cette fonction test est que son front de Pareto est discontinu. Elle a été utilisée pour mesurer la robustesse des algorithmes multi-objectifs à trouver ce front. La figure 5.8 montre les résultats d'une approche de type PESA, que nous avons implémentée et testée en utilisant les fonctions de bases de la plateforme [69]. En utilisant une population composée de 1000 individus, nous avons réalisé plusieurs simulations avec des taux de mutations variables. Les résultats montrent une sensibilité de cette approche à son paramétrage. Ainsi (figure 5.8), on s'aperçoit que pour certains taux de mutation l'algorithme est incapable de converger vers les solutions de Pareto de rang 0, et que le vrai front est obtenu pour un taux de mutation de 0.5. Ce taux mutation plus élevé permet d'améliorer l'exploration. Ces résultats montrent donc que certains paramétrages ne permettent pas une bonne exploration de l'espace des solutions en convergeant vers des minima-locaux.

En utilisant ce dernier front de Pareto nous avons pu comparer les résultats obtenus par cette approche avec les solutions fournies par l'algorithme que nous venons de décrire, figure 5.9. Ainsi nous avons associé un agent à chaque fonction et utilisé les capacités d'auto-organisation des agents pour réguler les contraintes, un agent architecte étant responsable de la modification des contraintes. Les résultats illustrent la capacité de notre approche à trouver de très bonnes solutions au problème, puisque nous trouvons des

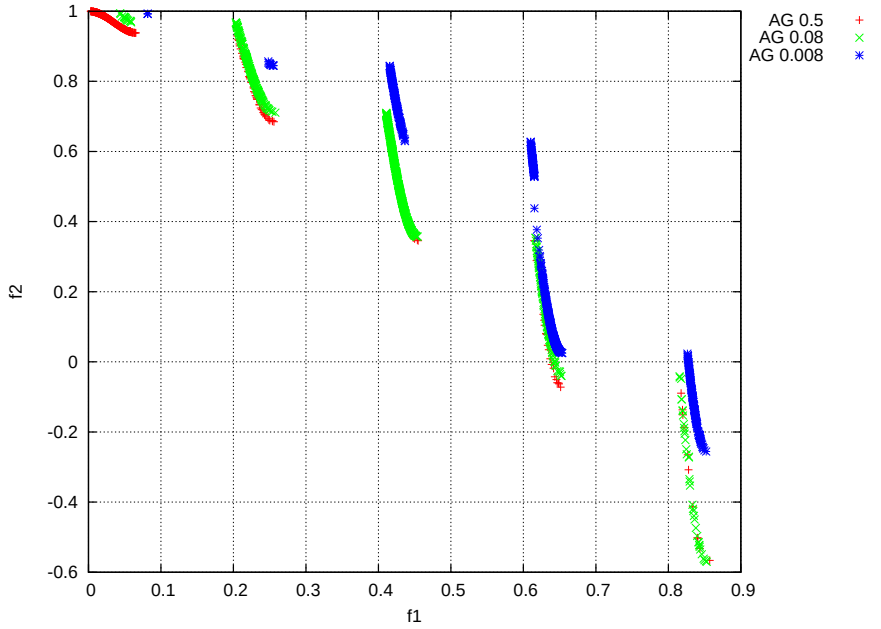


FIG. 5.8 – Fronts de Pareto pour différents paramétrages d'un algorithme génétique

points de rang 0 répartis sur chacune des parties du front. D'autre part, les résultats de MASCODE ont été obtenus immédiatement sans avoir à jouer sur le paramétrage de notre algorithme, car contrairement aux approches traditionnelles notre algorithme n'est pas basé sur le réglage d'une combinatoire, où il faut choisir entre l'exploitation des meilleures solutions identifiées et l'exploration de l'espace de recherche.

Néanmoins pour parcourir l'ensemble du front à l'aide de MASCODE, il a été nécessaire de partir de plusieurs points de départ afin de sauter certaines des discontinuités du front, que l'on appelle "*effets tunnels*".
D'autres modifications d'intervalles pourraient être étudiées afin d'éviter ces points de départs différents. Mais ces améliorations qui permettraient d'inciter les agents à aller vers ces espaces, augmenteraient aussi la dimension des espaces à parcourir. Finalement en considérant ce genre d'approches, on traite le problème en raisonnant globalement. Une bonne solution pourrait donc être de continuer à partir de différents points de départ, tout en choisissant ces points de départ d'une manière raisonnée.

5.5 Vers des approches dynamiques d'optimisation

Une tendance actuelle est l'étude des problèmes dynamiques, dont la formulation évolue au cours du temps [39] [13], car c'est le cas de la plupart des problèmes réels.

Ce besoin de dynamique impose de nouvelles contraintes fortes pour toutes les approches stochastiques car elles sont dépendantes de leur paramétrage, qui définit leur

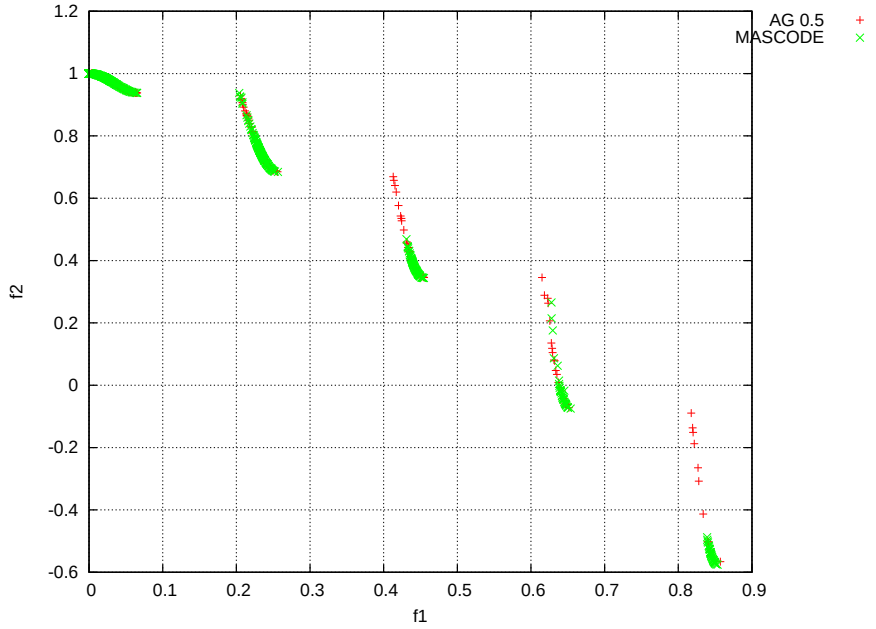


FIG. 5.9 – Résultats obtenus par MASCODE comparaison avec un AG

capacité à aller chercher de nouvelles solutions ainsi qu'à exploiter les solutions trouvées. Ainsi la plupart de ces approches convergent vers des bonnes solutions grâce à leur initialisation qui génère une population diversifiée puis à leur capacité à s'adapter en considérant les individus les plus efficaces dans l'environnement. En contrepartie, elles ont tendance à attirer tous les individus vers les solutions optimales au problème de départ. Ainsi lorsque la formulation du problème évolue, ces approches peuvent être incapables d'évoluer, car le génotype de la population s'est spécialisé vers une catégorie d'individus assez peu différents. L'étude de ces problèmes dynamiques a donc conduit à l'introduction de nouveaux mécanismes permettant la réintroduction de diversité au sein de la population, qui peut se faire de deux manières différentes :

- soit la population est capable de s'adapter seule au changement et ce grâce à des capacités propres, on parle alors d'approches *endogènes*.
- soit un mécanisme extérieur intervient pour modifier la population actuelle, on utilise alors des mécanismes *exogènes*.

5.5.1 Les approches endogènes

Les approches endogènes [10] sont basées sur le maintien de la diversité au sein de la population. Il peut se faire :

- **En conservant un facteur exploratoire assez important**, l'objectif est alors de conserver une population diversifiée en espérant qu'elle permette de converger plus

rapidement vers un nouvel optimum lors de la modification de la fonction objectif. Pour cela, on peut par exemple garder en permanence dans la population un nombre relativement important d'individus générés aléatoirement. Mais la génération aléatoire d'individus crée aussi un bruit important, qui peut avoir des conséquences sur les qualités de convergence de l'algorithme et conduire à une augmentation de la combinatoire.

- **En gardant en mémoire une partie d'individus non optimaux rencontrés au cours de la résolution**, ce qui permet d'assurer une variété génétique importante. Ce type d'approche est illustré dans [39], où la population est composée d'individus du front, d'individus maintenant une certaine diversité ainsi que d'individus permettant la prédiction de futurs optima obtenus en cas d'évolutions environnementales. Cependant la taille de la mémoire et le choix des individus non optimaux qu'il faut préserver est critique à la fois pour la convergence et la capacité d'adaptation de l'algorithme.
- **En utilisant des approches basées sur plusieurs populations recherchant des optima différents**, on espère conserver une diversité génétique en faisant évoluer plusieurs populations en parallèle avec des objectifs différents. Cette dernière approche est la plus intéressante car elle est discriminante. En générant des sous-populations relativement indépendantes, on peut espérer que toutes les sous-populations ne seront altérées par la modification de l'environnement. Mais pour être efficace, elle présuppose une bonne connaissance de l'espace de recherche nécessaire à la définition de sous-populations adaptées [10].

5.5.2 Les approches exogènes

Cette fois-ci, l'adaptation de la population provient de l'extérieur [10], c'est-à-dire qu'elle est déclenchée à la suite d'une modification environnementale (changement du problème). Dans ces approches, on retrouve des méthodes similaires aux précédentes avec :

- soit l'introduction de nouveaux individus choisis aléatoirement,
- soit la réintroduction d'individus moins bons rencontrés durant la résolution.

A priori ces méthodes devraient être plus intéressantes, que celles décrites précédemment puisque dans un premier temps elles n'interfèrent pas avec la recherche des solutions optimales. Mais finalement dès qu'il faut traiter une modification de l'environnement, le problème reste le même. En effet, la modification environnementale ne permet pas de savoir qu'elles sont les actions ou le paramétrage à réaliser pour obtenir une diversification de la population, qui soit suffisante à la recherche de nouvelles solutions.

5.5.3 Bilan et avantages de notre approche

D'autres approches similaires utilisant des algorithmes à base de particules ou à base d'agents ont été proposées, et présentés dans cette section. Mais en traitant les problèmes de manière globale, toutes ces approches sont limitées par la même contrainte conceptuelle, à savoir : comment concilier l'exploration et l'exploitation des solutions ?

À l'inverse, l'approche que nous proposons utilise la structure du problème et les relations entre les différents compromis pour le résoudre. De ce fait, elle a l'avantage d'être fortement discriminante, puisqu'une modification d'une contrainte dans l'environnement conduit automatiquement à une propagation locale de modifications et de compromis (d'agent en agent). La modification concerne alors uniquement certaines parties du système, ce qui permet une adaptation dynamique et rapide aux changements de l'environnement, et assure ainsi la convergence vers un autre point réalisant les nouvelles contraintes. Cependant de nouvelles optimisations autour de ce point doivent évidemment être réalisées afin de construire un nouveau front de Pareto.

5.6 L'intégration d'un raisonnement pour l'agent architecte

Pour améliorer ces optimisations autour des points de Pareto, nous pensons que notre algorithme pourrait utiliser un raisonnement basé sur une heuristique pour l'agent architecte. Ce raisonnement permettrait d'adapter les choix des sous-objectifs à optimiser en fonction des évolutions du système. Nous pensons qu'il serait par exemple intéressant d'améliorer l'ordre de sélection des sous-objectifs à optimiser en favorisant le choix du suivant selon ces critères :

- La distance du sous-objectif considéré à sa borne nominal est-elle importante ? Autrement dit, on cherche à savoir si la valeur de l'objectif est déjà optimisée.
- Le sous-objectif a-t-il une dépendance forte avec ceux déjà optimisés ? Les optimisations des sous-objectifs précédents ont-ils conduit à une optimisation ou à une dégradation de l'objectif considéré ?

Finalement, cette stratégie consiste à favoriser le choix des critères, qui sont les moins contraints et les moins interdépendants, et donc ceux pour qui l'optimisation a le plus de chance de réussir.

D'autre part et comme nous l'avons vu en appliquant notre approche à un cas test de référence (cf. section 5.4.2), le raisonnement de l'agent architecte pourrait aussi être amélioré pour choisir de nouveaux points de départ. En effet, lorsque le front de Pareto est discontinu il est parfois nécessaire de partir de plusieurs points de départs pour ob-

tenir toutes les parties du front. Cette fois-ci, l'agent architecte pourrait donc choisir de nouveaux points de départ, à partir de valeurs de paramètre de conception, qui n'auraient jamais étaient explorées et favoriser ainsi la recherche de nouveaux consensus. Là encore, il s'agit de le rendre plus efficace en favorisant la recherche de solutions sur de nouvelles portions de front.

5.7 Des approches interactives

La stratégie que nous avons proposée, présente une certaine généricité. Cependant, lorsque le nombre de degrés de liberté devient important et que les objectifs visés sont multiples, il devient difficile à la fois de parcourir l'espace des possibilités et de présenter l'information de manière à faciliter la prise de décision. Or l'interaction avec l'utilisateur sur les caractéristiques du problème (cf. section 2.1.4) repose sur sa compréhension des interdépendances entre les objectifs.

En offrant d'autres outils de visualisation, nous pensons que notre approche peut apporter d'autres moyens d'explications et d'interactions, qui puissent améliorer la compréhension du système et favoriser une co-construction des solutions avec le concepteur. Ainsi nous préconisons d'utiliser des approches décisionnelles interactives, qui permettent d'orienter le parcours de l'espace de recherche aux solutions intéressantes, et qui aident un concepteur à faire évoluer son problème en sachant par exemple : quelles sont les parties du système qui sont conflictuelles ; quelles sont les modifications qui lui permettront d'améliorer sa conception ; quels objectifs sur-contraignent le problème ; etc.

Comprendre est le commencement d'approuver.

Baruch Spinoza

6

La co-construction des solutions avec l'utilisateur

Sommaire

Les systèmes multi-agents offrent de nouveaux moyens conceptuels pour modéliser les systèmes. Néanmoins, la distribution de la connaissance qu'ils impliquent génère de nouveaux besoins en termes de pilotage et de supervision du système.
Selon la cybernétique, les systèmes de coopération Hommes-Machines doivent être considérés et décomposés selon quatre axes :

1. la décomposition en sous-éléments ;
2. la capacité des sous-éléments à agir ;
3. les moyens mis en place pour coordonner les actions des sous-éléments ;
4. l'intégration de la supervision humaine.

En général, le niveau 3 sert à détecter des problèmes liés aux décisions prises par le niveau 2, et le niveau 4 à intégrer les choix de l'utilisateur pour résoudre les problèmes

apparents. Dans MASCODE, la régulation du système pourrait être rattachée au niveau 2, et l'optimisation au niveau 3, puisque la régulation est basée sur des raisonnements purement locaux, et l'optimisation utilise une heuristique pour guider le comportement du système (cf. section 5.4).
Jusqu'à maintenant notre modèle permet de réguler des contraintes et de simuler un système complexe composé de modèles multidisciplinaires. Ces modèles composés d'une expertise humaine forte possèdent des propriétés et des caractéristiques qui ont leurs limites, et qui ne peuvent être réajustées que par un concepteur. Ainsi, le concepteur ne doit pas être extérieur à la résolution du problème mais participant dans sa conduite et sa supervision, ce qui implique une nécessité forte d'intégrer le niveau 4 dans notre approche.

6.1 Le cas de la conception avion

Tel que décrit en section 1.2.5, la conception avion requiert de nombreuses étapes, qui sont réalisées séquentiellement. La complexité des choix de conception implique souvent une redéfinition de certains objectifs du problème et donc une remise en cause de sa formulation initiale. Traditionnellement, cette phase de remise en cause nécessite la résolution d'un nouveau problème et une nouvelle itération des étapes 6, 7 et 8 de la figure 1.7 (cf. section 1.2.5). Une partie de ces itérations est due au manque de coopération entre le système de résolution utilisé et l'utilisateur. Ce manque de coopération relève de plusieurs aspects : premièrement du découpage du problème en étapes bien précises qui séquence la résolution ; deuxièmement du manque de capacités d'adaptation des outils de résolutions utilisés qui ne permettent pas à l'utilisateur de changer ses valeurs de critères au cours de la résolution.

Notre démarche de simulation distribuée, composée d'agents autonomes favorise la prise en compte de la complexité du problème, en offrant des capacités d'adaptation, et en résolvant les conflits à l'aide d'interactions entre les agents. Ainsi en tirant parti des propriétés du SMA et des situations rencontrées au cours de sa dynamique, on aide l'utilisateur à comprendre la structure et le comportement du système et on lui permet ensuite d'ajuster ses objectifs.

Cette assistance ne peut se faire qu'à la condition que le concepteur dispose d'outils lui permettant de coopérer avec le système, car cette démarche de co-construction centrée sur l'utilisateur favorise le partage des tâches avec le système. Dans l'approche proposée par P. Millot [57], deux modes de fonctionnement coexistent : un fonctionnement en mode de *"surveillance"* lorsque le système est en fonctionnement normal, et un mode *"pilotage"* lorsque le système rencontre des défaillances ou des situations pour lesquelles l'intervention humaine est nécessaire.

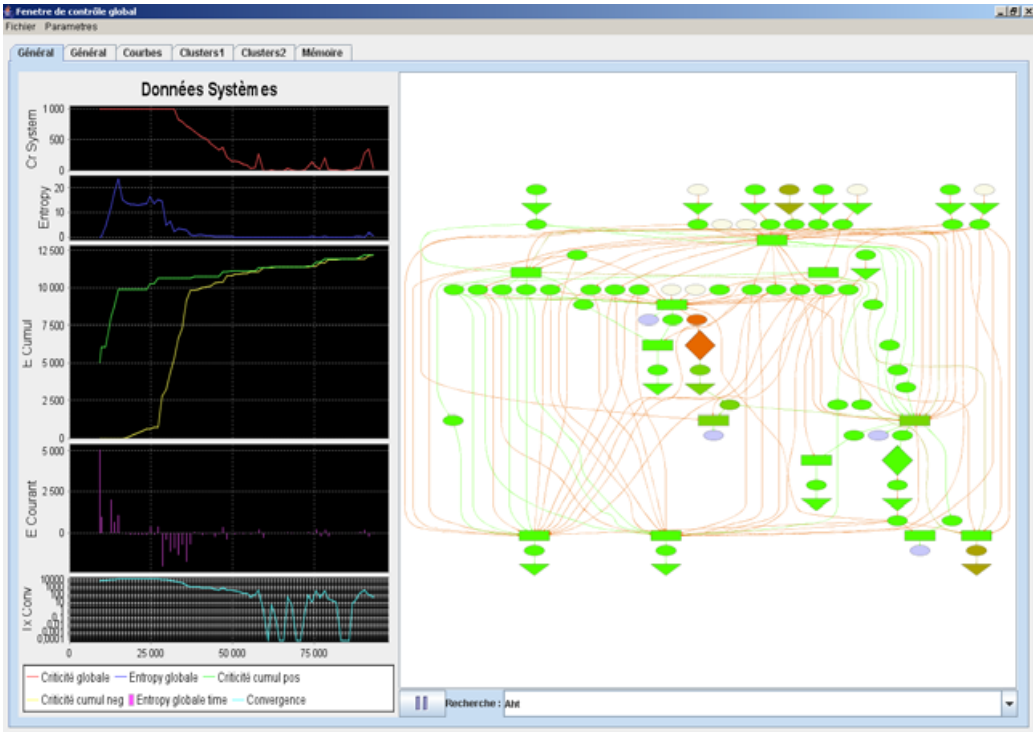


FIG. 6.1 – Vue principale sur le système

Comme nous voulons simuler le comportement des systèmes complexes, la partie surveillance est nécessaire pour restituer des informations comportementales à l'utilisateur et l'aider à piloter, lorsque cela est nécessaire.

6.2 La vue globale du système

Grâce aux comportements autonomes et auto-organisés de nos agents, le système est adaptatif et dynamique. Nous proposons d'utiliser une démarche coopérative et participative, qui doit éclairer le concepteur sur le mode de fonctionnement de ce système. Les interfaces utilisateurs doivent être conçues selon deux objectifs principaux : proposer une vue globale du comportement du système grâce à des observations globales, et permettre le pilotage et l'adaptation du comportement des agents à l'aide d'interfaces locales et propres à chaque agent.

6.2.1 La vue en temps réel

L'agent interface globale collecte des éléments globaux d'information. Une interface globale offre donc une vue graphique du système (figure 6.1), où tous les agents sont tracés sur un réseau avec un code de couleurs facilitant l'identification des agents les plus

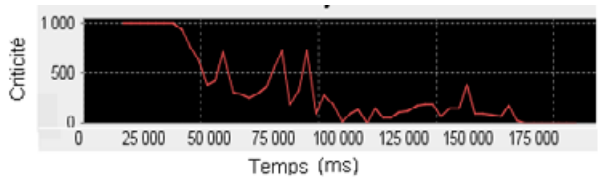


FIG. 6.2 – Criticité globale du système

insatisfaits. Ce réseau est dessiné à l'aide d'un graphe et fournit également une vision sur la manière dont les contraintes sont propagées dans le système en temps réel. Mais le comportement des agents est dynamique et la répartition des valeurs critiques en perpétuel mouvement. Le concepteur a besoin d'autres informations, comme des indicateurs qui seront détaillés par la suite pour comprendre l'état du système. Ainsi, il pourra anticiper et savoir si le système est dans une phase de convergence/stabilisation, ou bien dans une phase de résolution et de transfert d'énergie.

6.2.2 Les indicateurs de fonctionnement

En accord avec les principes de résolution de MASCODE, nous avons défini et intégré quelques indicateurs de fonctionnement au niveau global tels que :

- la criticité maximale du système ;
- l'absorption et la création de criticité dans le système ;
- la mesure de la dynamique du système.

La criticité maximale du système

Selon notre raisonnement coopératif, l'objectif local d'un agent est de diminuer la situation la plus critique de son voisinage. Chaque agent utilisant ce raisonnement, les situations critiques diminuent globalement à l'intérieur du système. Un bon indicateur sur l'état global du système est donc l'évolution de la situation critique maximale dans le système, présenté en figure 6.2. Mais cet indicateur n'est pas suffisant, car il ne rend pas compte de l'absorption ou de la création de désordre. Ainsi, la criticité globale du système peut globalement diminuer alors que la somme des criticités augmente. D'autre part, comme le système se compose d'interdépendances, en diminuant une situation critique dans une partie du système, on en génère souvent d'autres ailleurs. Par conséquent, la diminution de la criticité maximale est un phénomène discontinu (cf. figure 6.2), ce qui n'est pas toujours informatif sur l'état de la résolution. D'autres indicateurs plus robustes et moins sensibles aux changements instantanés du système sont donc nécessaires.

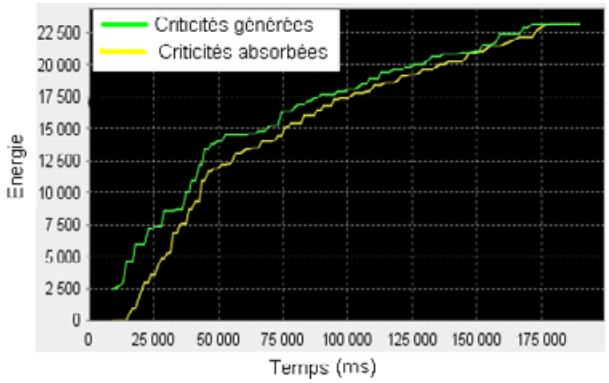


FIG. 6.3 – Cumul des absorptions/créations de criticité

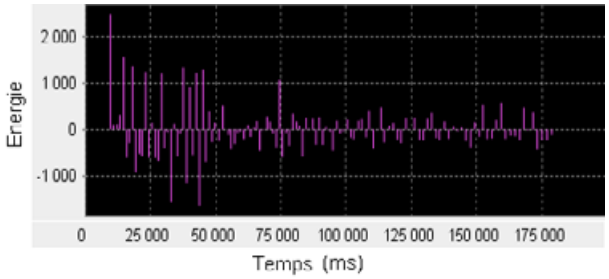


FIG. 6.4 – Transfert de criticité en temps réel

L'absorption et la création de criticité dans le système

À chaque réception d'informations locales, l'agent architecte additionne les variations des situations critiques à l'intérieur du système. Cette variation, tracée sur la figure 6.3 est positive, lorsque globalement des situations critiques ont été générées dans le système (une sorte de désordre), et inversement négatives lorsque globalement des situations critiques ont été absorbées. Finalement, on peut considérer que l'absorption et la génération de situations critiques correspond à une sorte de variation de l'énergie du système. Plus le système absorbe de l'énergie, en dissipant des situations critiques et plus il se rapproche d'un état stationnaire.

En observant les situations critiques absorbées et générées, tracées sur la figure 6.3, on remarque que de nombreuses situations critiques sont générées au démarrage du système, puis sont progressivement absorbées. Cette caractéristique est intéressante, elle montre que le système apprend les interdépendances progressivement et de manière indépendante à travers un processus d'essais et d'erreurs. Finalement, les premières modifications expérimentales de paramètres entraînent l'absorption de quelques situations critiques, mais en génèrent de nombreuses autres. Puis durant la résolution, des compromis intéressants sont identifiés et des actions qui réduisent les criticités de manière efficace sont choisies.

La mesure de la dynamique du système

Les différentes évolutions de la criticité du système (cumul, absorption, etc.) permettent de voir si globalement les niveaux de conflits dans le système sont en train de diminuer. Par contre, elles ne fournissent pas d'information sur la dynamique d'évolution des paramètres. Pour mesurer la dynamique d'évolution d'un paramètre, il faut prendre en compte deux choses :

la valeur du pas de modification du paramètre, nommée pas de modification : si le pas de modification est faible, le paramètre est en train de se stabiliser et donc sa valeur à peu de chance d'évoluer. À l'inverse un pas de modification élevé implique une dynamique importante.

la criticité des demandes de modification, si la criticité du paramètre et de ses demandes de modification sont élevées, c'est que sa valeur a des chances d'être modifiée.

Mais ces deux aspects sont complémentaires, nous avons donc choisi le paramétrage suivant pour mesurer *la dynamique d'un paramètre* :

$$\eta_{para} = \frac{\texttt{criticité-modification} * \texttt{pas-de-modification}}{\texttt{criticité-max} * \texttt{pas-de-modification-max}} \tag{6.1}$$

Avec ce paramétrage, on obtient les relations suivantes :

- si la demande de modification et le pas de modification sont élevés, le paramètre est en train d'évoluer fortement ; il est donc très dynamique ;
- si le pas de modification est important mais que le niveau de criticité est faible, c'est que la valeur du paramètre évolue mais qu'elle devrait converger rapidement ;
- si le pas de modification est faible mais le niveau de criticité élevé, c'est que les demandes sur le paramètre sont contradictoires et qu'il ne peut pas y répondre ; dans ce cas, sa valeur se stabilise sur un compromis ;
- si la demande de modification et le pas de modification sont faibles, alors le paramètre a peu de chance d'évoluer, puisqu'il n'en fera pas la demande.

Pour passer de la *dynamique d'un paramètre* notée η_{para}, à *la dynamique d'un agent* notée η_{agent}, on additionne simplement les mesures de dynamique de chaque paramètre d'entrée de l'agent :

$$\eta_{agent} = \sum \eta_{para} \tag{6.2}$$

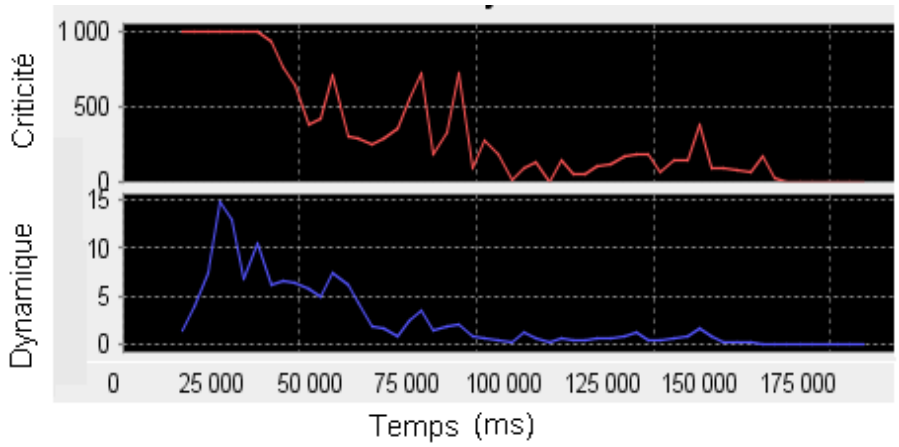


FIG. 6.5 – Mesure de la dynamique du système

Pour passer de *la dynamique d'un agent* à *la dynamique du système* notée η_{sys}, on additionne les mesures de dynamique des agents :

$$\eta_{sys} = \sum \eta_{agent} \tag{6.3}$$

L'évolution de la dynamique du système est présentée en figure 6.5. On remarque qu'elle est assez régulière et beaucoup moins sensible à des augmentations temporaires que la mesure de criticité du système. Dans un premier temps, la dynamique du système augmente, cela correspond à la phase d'initialisation du système, puis elle diminue progressivement, c'est la phase de résolution puis de stabilisation et de convergence.

6.3 La vue agent et la résolution de conflit

Les agents sont intrinsèquement subjectifs et incertains au sujet des conséquences de leurs actions, mais de leurs actions individuelles émerge un système adaptatif. Ainsi, l'incertitude et la subjectivité des décisions ne doivent pas être vues négativement, comme une perte de l'ordre, mais comme un facteur de créativité, d'adaptation et d'évolution.

Avec la surveillance globale du système, les concepteurs connaissent quelles parties du système sont critiques, et suivent les indicateurs sur l'état global du système. Mais ces informations ne sont pas suffisamment détaillées pour permettre à l'utilisateur d'identifier les raisons d'un conflit ou d'avoir une certaine compréhension du mode opératoire qui a conduit le système vers un état d'équilibre particulier. Ainsi ce mode de représentation ne suffit pas à indiquer à l'utilisateur les modifications qu'il peut réaliser pour soit optimiser les solutions, soit résoudre des conflits. Or pour la gestion des interdépendances et des aspects multi-objectifs, il est primordial que le concepteur trouve quelles sont les

contraintes à changer.

6.3.1 La mémoire de l'agent à court terme

L'interface permettant d'avoir une vue globale, montre les agents critiques au sein du système à un instant donné. Ce mode de représentation ne permet pas d'identifier pourquoi certains de ces agents se sont mis dans un état critique. Prenons un exemple :

- des agents objectifs de sorties sont critiques ;
- ils génèrent des demandes de modification, qu'ils envoient aux agents qui leur fournissent les valeurs de paramètres ;
- ces demandes sont propagées d'agent en agent et finissent par cheminer, jusqu'aux agents gérant les paramètres entrées du système ;
- lorsque des demandes de modification sont conflictuelles, les paramètres en entrées sont modifiés de manière à converger vers un consensus équivalent pour chacun des conflits ;
- ainsi si le système est sur-contraint, il se stabilise autour d'une valeur.

Dans ce cas de figure, la vue système présente les agents dont la situation ne peut être améliorée, (cf. figure 6.1). Cette vue ne permet pas d'identifier la manière dont les paramètres sont en conflits et donc de proposer des modifications de contraintes.

Pour proposer ces alternatives ou une information sur les paramètres en conflit, chaque agent mémorise le chemin des dernières requêtes qu'il a reçues. Ainsi, l'utilisateur peut voir le sens des demandes de modifications reçues par un agent, comprendre le cheminement des requêtes et ainsi identifier les paramètres en situations conflictuelles à l'aide d'un graphe de conflits (cf. figure 6.6). Cette figure présente des demandes de modifications conflictuelles pour le paramètre *span* (envergure). Les rectangles représentent les agents à l'origine de requêtes ou par lesquelles elles ont cheminées ; les traits pleins les demandes positives et les traits pointillés les demandes négatives. Ainsi, on s'aperçoit que des demandes positives ont été adressées pour les paramètres RA (rayon d'action) et vz_{clb} (vitesse de montée), et des demandes négatives pour $MTOW$ (masse) et Ar (ratio envergure-fuselage). Dans ce cas, la résolution du conflit pourra être résolue en donnant plus de degrés de liberté aux contraintes et objectifs en conflit. On pourra, par exemple, augmenter les valeurs limites de $MTOW$ et de Ar et aussi supprimer les demandes de modifications induites actuellement pour ces paramètres.

6.3.2 La mémoire de l'agent à long terme

Les chemins conflictuels représentent la mémoire des conflits à court terme et permettent de résoudre la plupart des conflits. Cependant cette mémoire n'est pas toujours

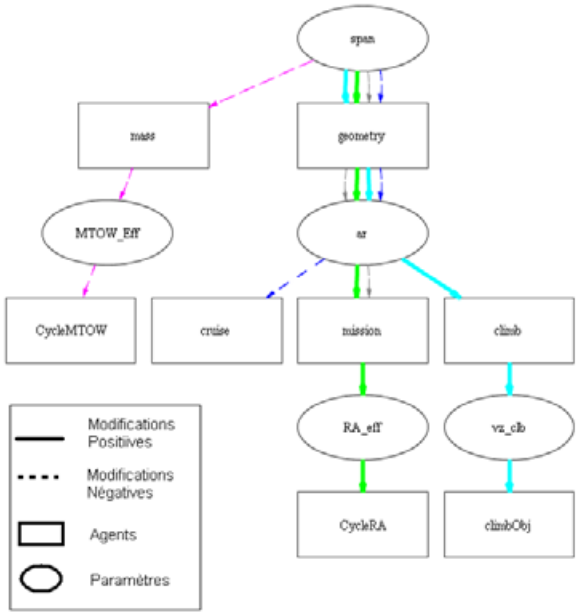


FIG. 6.6 – Requêtes en conflit

suffisante, car non représentative de l'ensemble des modifications réalisées par un agent. Ainsi elle ne fournit pas d'information sur l'ensemble des relations qu'a une contrainte avec les autres. Pour améliorer la compréhension des interdépendances, une autre mémoire cumule les modifications réalisées sur une entrée durant le processus de résolution, en les classant par origine. Ces modifications cumulées sont ensuite tracées sur un histogramme vertical (cf. figure 6.7). Par la suite, si le concepteur décide de changer la valeur de la contrainte de ce paramètre d'entrée. Il pourra anticiper quels pourront être les paramètres objectifs de sorties impactés par sa modification.

Pour résumer, la surveillance globale du système fournit une vue d'ensemble. Les interfaces locales d'agent permettent au concepteur d'entrer à l'intérieur du système et d'analyser une partie de la façon dont le système a convergé. Ces interfaces locales peuvent également prendre en compte à tout moment des modifications du concepteur, puisque les contraintes et la connaissance sont définies localement.

6.4 Vers une approche coopérative et multi-concepteurs

Au-delà des mécanismes de régulations et des connaissances sur les modèles évoqués dans la partie précédente, la plupart des systèmes complexes possèdent plusieurs niveaux de décomposition et d'abstraction. En conception avion, on commence par concevoir l'avion en utilisant des outils de simulation à gros grains avant de descendre sur des niveaux fins de précision. Cette décomposition pose évidemment de nombreuses questions

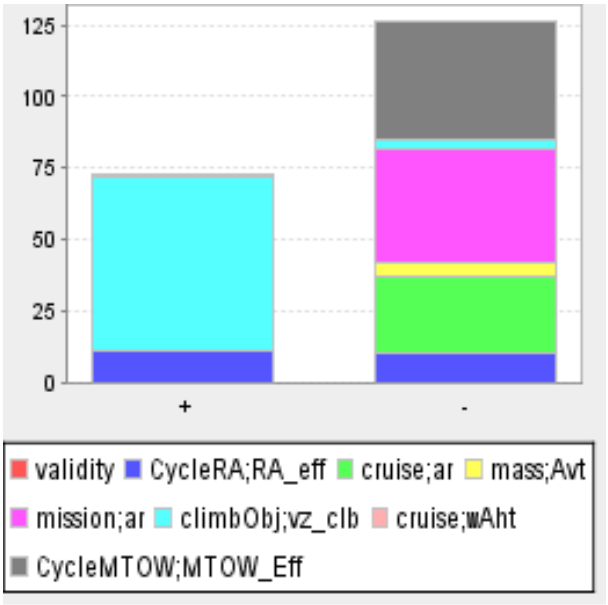


FIG. 6.7 – Cumul des modifications d'une entrée par origine de requête

lorsque l'on s'intéresse à la simulation et à la supervision de tels systèmes :

- Concernant la modélisation de l'outil de simulation, comment peut-on intégrer des aspects multi-niveaux ?
- Concernant les interactions entre l'outil et ses utilisateurs :
  - Comment établir des coopérations avec des utilisateurs aux connaissances et aux besoins variés ?
  - Comment proposer des outils d'aide à la décision qui intègrent plusieurs niveaux de décision ? Certains auraient des compétences très précises sur un domaine, d'autres une vue générale mais complète du système.

Ces différents aspects impliquent la prise en compte d'autres outils de supervision qui intègrent des besoins propres à chaque concepteur : leur perception environnementale, leur mode de décision, etc. L'évolution de notre approche vers des modèles multi-concepteurs nécessite donc l'ajout de nouvelles coopérations Homme-Machine utilisant [56] :

- des modèles cognitifs : permettant d'intégrer la perception environnemental de l'utilisateur disciplinaire ;
- des modèles normatifs : permettant de décrire les propriétés exigées plus formellement ;
- des modèles prédictifs : illustrant le comportement dynamique du système ;
- des modèles de pilotages : aidant au pilotage et à la décision.

En réfléchissant à l'ensemble de ces aspects, on franchit une nouvelle étape dans l'intégration de notre système vers des approches multi-utilisateurs. L'intérêt est alors d'utiliser

les capacités de raisonnement et d'abstraction de MASCODE pour aider les concepteurs à s'approprier les contraintes manipulées par d'autres concepteurs en utilisant les capacités de raisonnement, de communication et de coordination d'activités des agents. En intégrant la perception de chaque discipline, on se place dans une approche de simulation, qui pourrait répondre à des interrogations provenant des concepteurs en les aidant à percevoir les contraintes manipulées par autres.

Synthèse de partie

Dans cette partie, nous avons proposé un algorithme de régulation de contraintes basé sur une approche auto-organisatrice, pour laquelle chaque agent utilise un ensemble d'actions coopératives et locales. Ainsi, la résolution du problème est conceptuellement distribuée et adaptative aux changements environnementaux.

En utilisant les capacités adaptatives du système, un agent architecte modifie les caractéristiques environnementales des agents en renforçant certaines contraintes sur les objectifs, ce qui a pour effet de perturber le système. De ce fait, il permet un déplacement du système dans l'espace des solutions. A l'aide d'une stratégie de modification adaptée, on peut alors définir des déplacements vers des zones préférées et réaliser ainsi l'optimisation de certains objectifs. En mémorisant les solutions non-dominées rencontrées, l'agent architecte est capable de construire un front de Pareto en contraignant les agents disciplinaires du niveau inférieur.

L'optimisation des solutions est importante, car elle répond à des attentes des concepteurs et nous permet de nous confronter aux techniques traditionnellement utilisées. Cependant cette manière de répondre au problème initial ne permet pas d'exploiter toutes les caractéristiques de notre approche, puisqu'elle ne fournit pas d'information sur l'état de nos agents, sur la nature des conflits rencontrés/résolus, etc. Nous avons donc fourni :

- des indicateurs permettant de suivre l'état de la résolution ;
- des interfaces favorisant la détection et la résolution de conflits ;
- des moyens pour suivre l'état des agents et modifier les caractéristiques du problème.

La figure 8 résume les solutions proposées :

- les agents disciplinaires représentent le coeur de notre approche. Ils permettent la régulation des contraintes ;
- l'agent architecte collecte des informations provenant des agents disciplinaires et modifie l'environnement pour optimiser certains objectifs ;
- des interfaces construites à partir d'informations collectées par l'agent architecte permettent au concepteur de suivre l'état général de la résolution ;
- des interfaces attachées à chacun des agents l'informent sur les conflits rencontrés au niveau disciplinaire, et lui permettent de modifier ses contraintes s'il le souhaite.

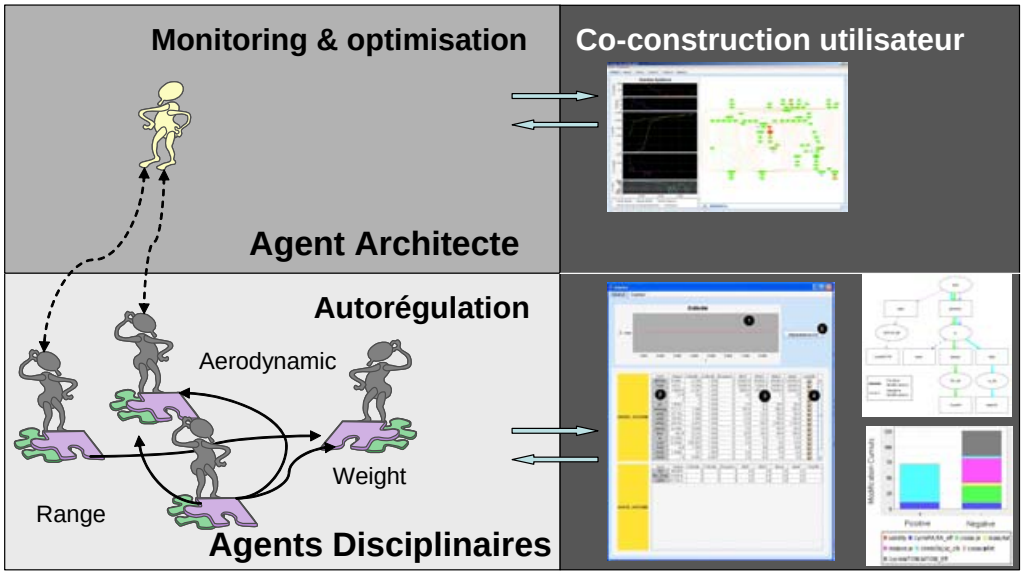


FIG. 8 – Récapitulatif des solutions proposées

Troisième partie

Application des propositions à la conception préliminaire avion

Les lois claires en théorie sont souvent un chaos à l'application.

Napoléon Bonaparte

7 L'application MASCODE

Sommaire

Dans ce chapitre, nous présentons la structure des modèles disciplinaires que nous avons utilisés ainsi que quelques choix d'implémentation pour notre démonstrateur MASCODE.

7.1 Vers une ingénierie des modèles

La durée de vie des produits aéronautiques et la diversité des métiers impliqués dans la conception rendent l'ingénierie des modèles difficiles. Souvent les modèles sont développés et utilisés pour des besoins spécifiques (certification, nouvelles technologies, etc.). De ce fait, ils sont souvent hétérogènes, attachés à un contexte et particulièrement difficiles à maintenir et à réutiliser.

La contextualisation des modèles ainsi que la collecte d'informations améliorent leur cycle de vie et leur réutilisation, et ce sont donc des sujets tout à fait d'actualité pour les constructeurs aéronautiques. Pour combler ces lacunes, des projets Airbus tels que CVB (Common Virtual Bird) ou TOPCASED (Toolkit in OPen-source for Critical Application and SystEms Development) participent à l'ajout de connaissances, d'outils et de méthodes améliorant ainsi l'ingénierie des systèmes avions. Par exemple, à travers ces projets, des outils d'ingénierie système sont promus et mis à la disposition des ingénieurs afin de faciliter l'utilisation de langages de modélisation ciblant différents aspects du processus de développement. La finalité est de mettre à disposition de nouveaux moyens favorisant la gestion de connaissances, i.e. par la prise en compte des spécificités de chaque composant, la formalisation des données dans des langages de modélisation, ou encore la définition d'exigences fonctionnelles.

Dans ce contexte, il est donc pertinent de développer des approches qui favorisent la prise en compte de connaissances associées aux modèles manipulés.

Aux avant-projets, un modèle de simulation a un certain nombre de caractéristiques. Il possède et échange des paramètres, ce sont ses entrées/sorties. Chaque paramètre a une unité (mètre, pied, etc.) ainsi qu'un domaine de validité souvent associé à un contexte de simulation nominal. En fonction des connaissances sur le domaine de validité, de la valeur du paramètre fourni et de la nature du modèle, on est aussi parfois capable de définir des mesures d'incertitudes et donc de qualifier la précision du modèle.

Il existe trois types différents :

- soit ils servent à réaliser des simplifications géométriques ;
- soit ils exécutent des calculs physiques, tels que des calculs de masses, des calculs de missions ou de distance au décollage ;
- soit ils établissent des performances telles que la consommation par passager et le coût d'exploitation (*direct operating cost*, *maintaining cost*).

En accord avec ces propriétés, un modèle est donc utilisé pour la résolution de certains problèmes, puisqu'il a été conçu en tenant compte d'un contexte particulier. Ainsi selon ses besoins de simulation, un concepteur est capable de choisir parmi des alternatives de modèles, celui qui lui semble le plus adapté à son contexte, en tenant compte notamment de leurs plages de validités. Une fois les modèles choisis, des phases d'ajustement (calibrages) sont réalisées telles que décrit en section 1.2.5. Mais la considération de tous ces aspects est assez récente, et il existe encore assez peu d'ingénierie des modèles et de réflexion sur les conséquences des réutilisation/re-calibrages successifs de modèles conçus pour une utilisation initiale différente.

Finalement, notre démarche qui consiste à tenir compte des connaissances relatives à ces modèles en les associant en local et en les utilisant lors de la résolution du problème

comme des connaissances propres aux agents, permet donc d'aller dans le bon sens.

7.1.1 La contextualisation et les aspects multi-niveaux de la conception avant-projet

En plus des connaissances associées aux modèles, il existe trois niveaux de modélisation différents :

- le premier niveau de granularité permet de faire des estimations de performances à un très haut niveau d'abstraction. Dans ce cas de figure, les modèles utilisés sont basés sur des interpolations construites à partir des performances d'avions existants ;
- le second niveau se base sur des modèles représentant la physique simplifiée générale avion. Dans ce cas, il s'agit de partir de lois simples pour estimer les performances de l'avion. Mais des méthodes de calibrage basées sur des données d'avions existants, sont néanmoins utilisées afin d'adapter ces lois simples à une certaine réalité physique ;
- le troisième niveau de granularité utilise des modèles plus précis provenant des métiers (discipline). Ce troisième niveau permet de confirmer/raffiner les choix de conception qui ont été faits lors de l'étape précédente, en utilisant des outils d'évaluation qui sont adaptés à un niveau de définition plus détaillé.

Dans le cadre de nos travaux, nous ne tenons donc pas compte de la granularité, ni des choix de modèles pour construire la fonction de simulation (choix organisationnels), et ceci pour deux raisons :

1. manque actuel de modèles qualifiés avec des domaines de validité, permettant d'automatiser leur mise en concurrence en fonction du contexte de simulation ;
2. manque actuel de possibilités pour faire communiquer des modèles de simulation appartenant à des niveaux différents de conception (détail).

L'ensemble des expérimentations a donc été réalisé en utilisant les modèles de niveau 2, qui correspondent à ceux de l'USMAC (*Ultra Symplified Model Aircraft*), présenté ci-dessous.

7.1.2 Présentation de l'USMAC - *Ultra Symplified Model Aircraft*

L'USMAC a été créé pour les besoins spécifiques de la conception avant-projet. Ce modèle a l'avantage d'être représentatif de la complexité multi-disciplinaire du problème, tout en utilisant des calculs peu coûteux en temps. De ce fait, des simplifications ont été réalisées sur les modèles, et non sur leurs relations avec les autres. Finalement, l'USMAC

Fuselage data	Npax	Nombre de passagers
	NpaxFront	Nombre de passager par rangée
	Naisle	Nombre de couloir
Propulsion data	FNslst	Puissance moteur
	BPR	Engine bypass ratio
	ne	Nombre de moteur
Wing geometry	Awing	Aire de ailes
	phi	Angle d'attaque
	span	Envergure
	tuc	Corde
Flight conditions	disa	Delta ISA
	mach	Mach
	alt	Altitude

TAB. 7.1 – Paramètres d'entrées de l'USMAC

se caractérise par un ensemble de fonctions de Scilab [INRIA, 1989], qui répond aux conditions suivantes :

1. capturer les aspects multi-disciplinaires de la conception ;
2. être simple et rapide en termes d'exécution ;
3. s'adapter facilement à différents types de données (selon le problème).

Pour le moment, l'USMAC est trop simple pour être suffisant aux besoins d'une étude industrielle. Mais, c'est un outil bien adapté pour expérimenter et valider de nouvelles approches mathématiques et informatiques qui permettent d'améliorer la conception avion.

Les entrées

L'USMAC est composé de 12 entrées, qui permettent de définir le fuselage, la propulsion, la géométrie et les conditions de vol, tableau 7.1.

Les sorties

L'USMAC permet de calculer une dizaine de performances, tableau 7.2

Les fonctions

L'USMAC est composé d'une centaine de fonctions basiques. Chacune de ces fonctions basiques appartient souvent à une fonction disciplinaire de plus haut niveau. Par exemple, l'évaluation de la masse d'un avion est composée des fonctions de bases qui permettent d'évaluer la masse des différents composants de l'avion (empennage, ailes, fuselage, train

Vapp	Approach speed	Vitesse d'approche
TOFL	Take-Off Field Length	Distance nécessaire au décollage
Kfn	Cruise thrust	Facteur de confiance sur le moteur
Vz	Climb speed	Vitesse de montée
Fff	Fuselage fuel ratio	
MTOW	Maximum Take-Off Weight	
OWE	Operational Weight Empty	Masse de l'avion à vide
PL	Payload	Capacité de transport
Fuel	Fuel	quantité de carburant
RA	Range	Rayon d'action de la mission

TAB. 7.2 – Performances de l'USMAC

d'atterrissage et moteur). Ces modèles basiques ont été construits en utilisant différentes hypothèses et peuvent être regroupés en deux catégories principales. Certains modèles sont construits à l'aide de grandeurs physiques ou de simplifications géométriques, alors que les autres sont issus de lois physiques et/ou de régressions statistiques établies à partir des bases de données d'avions existants.

Voici, quelques exemples de modèles :

1. **Un modèle géométrique** : la fonction *fuselage length* fait partie du domaine disciplinaire *geometry*. Elle permet de calculer la longueur du fuselage à partir du nombre de passagers ($Npax$), du diamètre du fuselage ($dfus$) et du nombre de passagers par rangée ($NpaxFront$).

```
function lfus = fuselage_length_(Npax,NpaxFront,dfus)
//================================
// Model: geometry
// Constants : c1, c2, c3
lfus = c1 * dfus + c2 * Npax/NpaxFront + c3;
endfunction
```

2. **Un modèle de masse** : la fonction *fuselage mass* fait partie du domaine disciplinaire *weight estimation*. Elle estime la masse du fuselage ($Mfus$) en connaissant son diamètre ($dfus$) et sa longueur ($lfus$).

```
function Mfus = fuselage_mass_(dfus,lfus)
//===========================
```

```
// Model: weight
// Constants : c1, c2, c3
sfus = %pi*dfus*lfus ;
Mfus = ( c1 * sfus - c2 ) * sfus + c3 ;
endfunction
```

Pour réduire la complexité du réseau de fonctions, ces fonctions de bases sont encapsulées dans des fonctions de domaines aux dimensions plus conséquentes (*geometry*, *weight estimation*, *approach speed*), qui servent en général à calculer des performances. Voici, par exemple, une fonction qui permet d'estimer la vitesse d'approche de l'avion (*vapp*) :

```
function [vapp] = approach_speed_(alt_app,LDW,aircraft_data)
//non_scalar:aircraft_data
//===============================================================
global UF
//---------------------------------------------------------------
[Aref,phi] = aircraft_data(['Aref','phi']) ;
kvs_LD = Kvs_Landing_() ;
g = gravity_acc_(alt_app) ;
[KczmaxLD] = Cz_max_LD_factor_(UF.KczmLD) ;
[czmax_LD] = Cz_max_LD_(KczmaxLD,phi) ;
vapp = app_speed_(LDW,czmax_LD,Aref,g,kvs_LD) ;
endfunction
```

Le bouclage Masse-Mission

Pour calculer la masse maximale au décollage d'un avion ($MTOW$), on devrait additionner simplement la masse de ses composants principaux (les ailes M_{Wing}, le fuselage M_{Fus}, le train d'atterrissage M_{Gear}, etc.). Mais les méthodes réalisant ces estimations sont elles-mêmes basées sur des estimations de la masse maximale au décollage. Ceci introduit des boucles dans la fonction globale et implique la résolution du bouclage Masse/Performances, tel que décrit en section 1.2.4. Dans l'exemple traité, une seconde boucle co-existe concernant le calcul de la mission. Ainsi, on peut avoir, par exemple, le phénomène suivant : une augmentation du rayon d'action (RA) augmente la quantité de fuel nécessaire, ce qui augmente la masse au décollage et donc ceci implique une diminution du rayon d'action.

```
Augmenter RA => Augmenter Fuel => Augmenter MTOW =>
Augmenter la consommation => Diminuer RA.
```

Déterminer la masse au décollage et le rayon d'action sont donc des points cruciaux de la conception avant-projet.

7.1.3 Bilan

Nous avons donc utilisé les fonctions de domaine du second niveau de granularité. En assemblant ces fonctions, on obtient ainsi un système composé de modèles disciplinaires, de boucles, de paramètres objectifs en entrée et en sortie. En encapsulant ces modèles dans une architecture d'agents MASCODE, on construit un système capable de poser et de résoudre des problèmes de conception préliminaire avion.

7.2 L'implémentation de MASCODE

7.2.1 Les besoins fonctionnels

L'implémentation de notre système multi-agent devait répondre à quelques exigences très simples, pouvoir utiliser les fonctions métiers écrites en Scilab, en se basant sur une architecture conceptuellement distribuée et donc à base de *threads*. Pour limiter les contraintes liées à la distribution du système, nous avons choisi d'utiliser une plateforme agent. En plus de la distribution du système, nous avons voulu tester l'utilisation d'une plateforme sur un cas d'application industriel. Ainsi nous avons choisi d'utiliser la plateforme JADE [7] pour sa généricité, sa maintenabilité et son utilisation grandissante.

7.2.2 La plateforme JADE

La plateforme JADE permet de faire abstraction d'un certain nombre d'éléments. Elle met à disposition des agents de gestions permettant de gérer des annuaires, des bus d'échanges de données ainsi que l'adressage des messages de communication. De ce fait, il est très facile de distribuer les agents sur un réseau et d'échanger des informations à l'aide de messages.
La conception de chaque agent est également facilitée grâce à l'utilisation de comportements (*behaviours*) qui peuvent être créés dynamiquement, exécutés en parallèle, en série, etc. Pour une description détaillée de JADE, nous renvoyons le lecteur au site internet du projet : `www.jade.tilab.com`

7.2.3 Les agents plateformes

JADE simplifie la distribution des agents, puisqu'il suffit d'utiliser les capacités de la plateforme. La distribution est d'autant plus simple et intéressante à réaliser, que notre

modèle d'agent est conceptuellement distribué. Pour distribuer le système, un agent plateforme a donc simplement besoin d'être créé sur une machine distribuée et de s'enregistrer à un agent "page-jaune" (*YellowPage*) sur la plateforme principale JADE. L'adressage physique des messages et leur envoi est ensuite assuré par les agents de JADE.

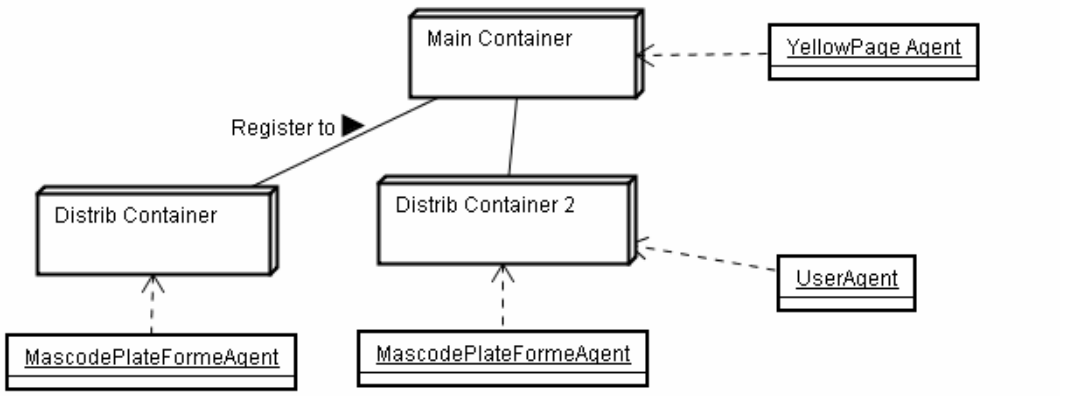


FIG. 7.1 – Modèle de répartition

7.2.4 Le lancement du système MASCODE

Pour lancer une application MASCODE, il suffit d'exécuter une plateforme JADE sur un ordinateur du réseau, puis de créer des agents plateformes MASCODE sur une ou plusieurs machines du réseau, qui s'enregistrent à la plateforme JADE. Lorsque ce support est initialisé, on crée un cas test à l'aide d'un agent utilisateur MASCODE (*UserAgent*), qui lance une résolution en utilisant le processus d'exécution suivant et illustré en figure 7.2 :

1. l'agent utilisateur interroge l'agent page-jaune de JADE sur la disponibilité d'agents plateformes MASCODE dans le réseau ;
2. l'agent page-jaune de la plateforme JADE répond à cette requête ;
3. en utilisant la liste d'agents plateformes disponibles, l'agent utilisateur (*UserAgent*) distribue aléatoirement ses modèles de simulation sur le réseau ;
4. il récupère les adresses de l'ensemble des agents ainsi créés et informe les agents échangeant des paramètres, des adresses de leurs partenaires ;
5. une fois les agents informés de leurs partenariats, l'agent utilisateur crée l'interface graphique utilisateur pour le suivi du système ;
6. par la suite des échanges entre agents partenaires sont réalisés uniquement à l'initiative des agents disciplines ;

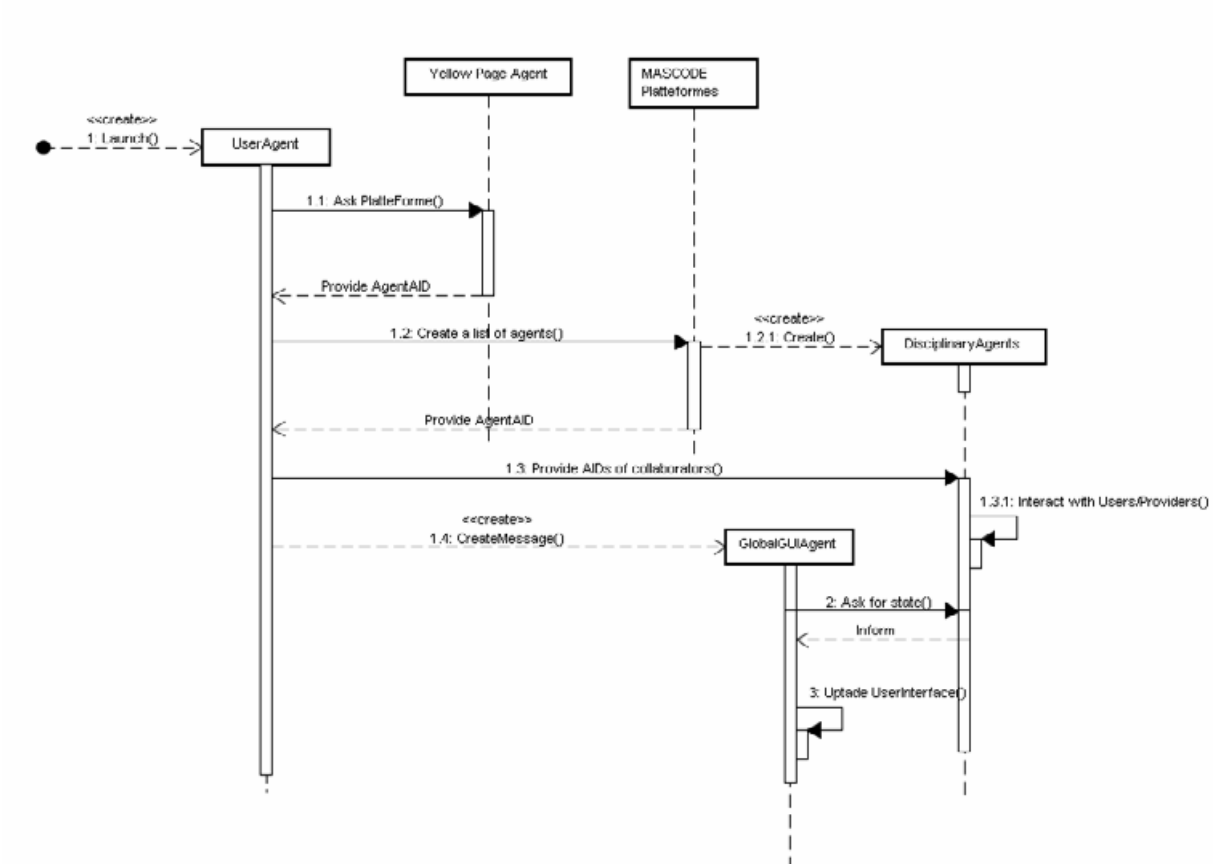


FIG. 7.2 – Séquence de lancement de l'application

7. afin de suivre, l'état du système, l'agent "architecte" qui joue aussi le rôle d'agent utilisateur (*UserAgent*), interroge régulièrement chaque agent discipline sur son état afin de mettre à jour la vue globale de l'état du système.

On fait la science avec des faits, comme on fait une maison avec des pierres : mais une accumulation de faits n'est pas plus une science qu'un tas de pierres n'est une maison.

Henri Poincaré

8

Les résultats et les performances de MASCODE

Sommaire

En utilisant les agents décrits dans le chapitre 7 ainsi que la modélisation générale introduite dans la partie II, MASCODE a été implémenté. Des expériences ont permis de vérifier ses propriétés et ses performances. Certains de ces résultats sont comparés à d'autres approches, lorsque cela est possible. Mais l'implémentation de MASCODE étant originale certaines de ses propriétés lui sont propres, la présentation de ces propriétés différenciantes fait l'objet du début de ce chapitre.

8.1 Les résultats propres à MASCODE

8.1.1 La convergence vers un état satisfaisant les contraintes

Pour expérimenter notre approche, nous avons utilisé l'USMAC (cf. section 7.1.2). Cette première expérimentation se base sur un cas test composé de 10 modèles et de 60

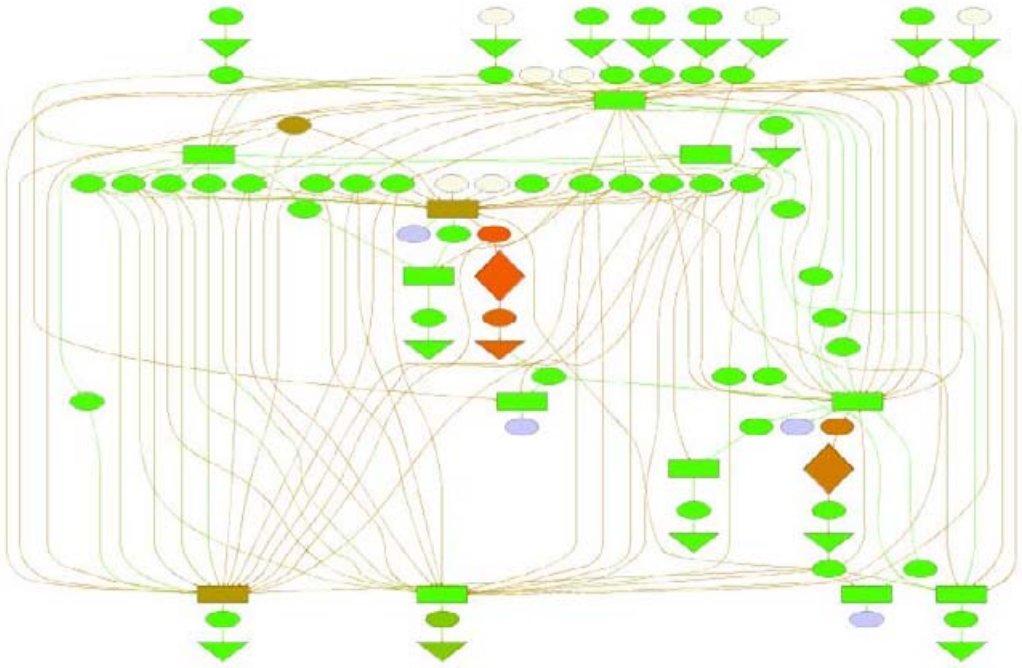


FIG. 8.1 – Cas test composé de 30 agents

paramètres (20 entrées, 17 sorties, 23 paramètres intermédiaires). Parmi ces paramètres, certains sont des objectifs (7 entrées, 7 sorties), c'est-à-dire que l'utilisateur a proposé des intervalles objectifs pour ses paramètres. Pour rappel, les objectifs sur les entrées représentent des contraintes sur les paramètres de conception de l'avion, alors que les contraintes sur les sorties représentent des critères de performances.

Cette étude est ensuite agentifiée en utilisant, la transformation présentée en section 4. De ce fait des agents boucles de rétroactions et objectifs sont ajoutés ; le système complet est représenté en figure 8.1. Il est composé de 30 agents et de deux boucles.

En utilisant l'algorithme de régulation présenté en section 4.3.8, les agents négocient les valeurs de paramètres de conception. Les résultats de cette négociation sont présentés en figure 8.2, qui montre l'évolution des paramètres de conception et de performances. L'axe des abscisses représente le temps, l'axe des ordonnées les valeurs normalisées de chaque paramètre. La figure 8.3 illustre les valeurs critiques de l'ensemble de ces mêmes paramètres, alors que la figure 8.4 présente l'évolution de la valeur critique du système dans son intégralité au cours du temps.

Le système trouve une solution, lorsque les valeurs critiques sont nulles. On s'aperçoit que la valeur critique du système diminue pour converger vers une première solution à $t = 90s$. Durant cette première phase (t compris entre 0 et $100s$), les valeurs des para-

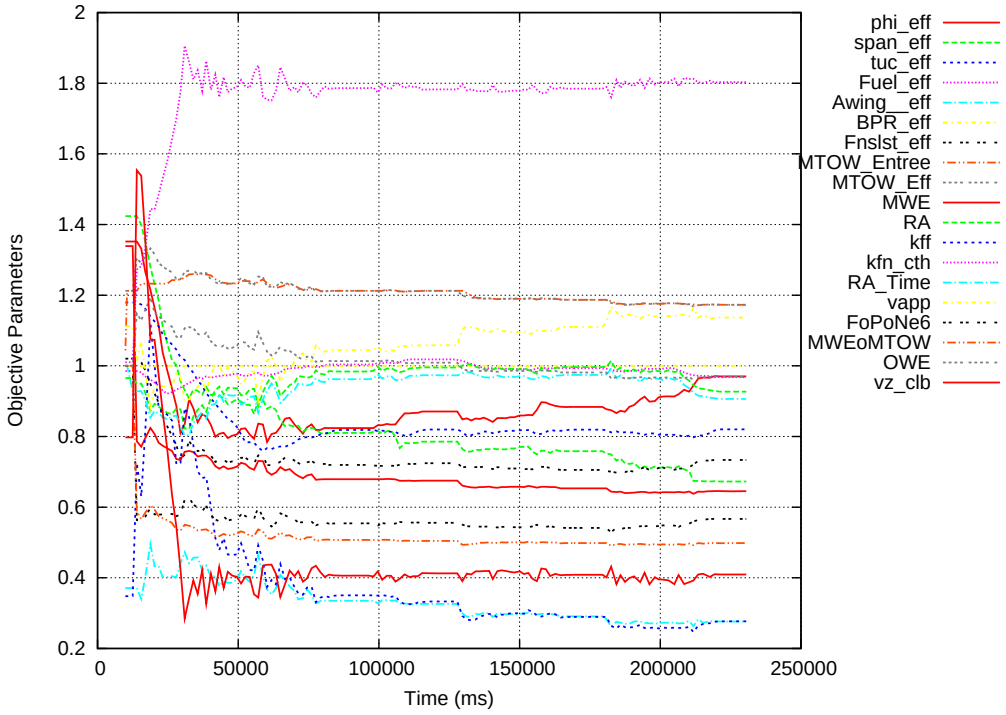


FIG. 8.2 – Évolution des valeurs des paramètres objectifs durant une exécution

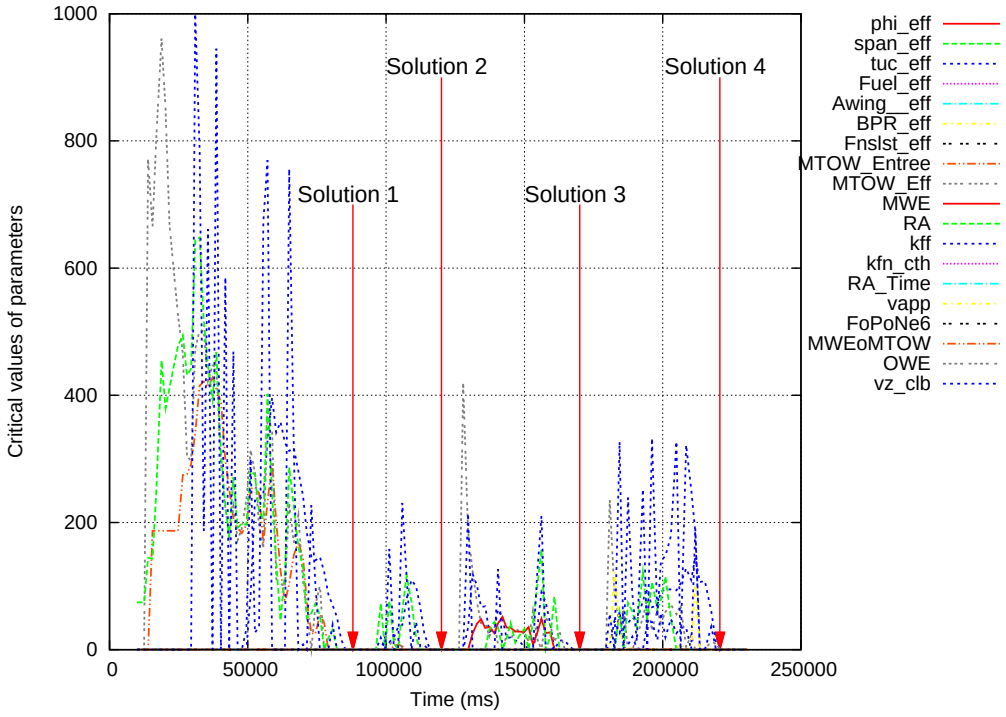


FIG. 8.3 – Évolution des valeurs critiques des paramètres

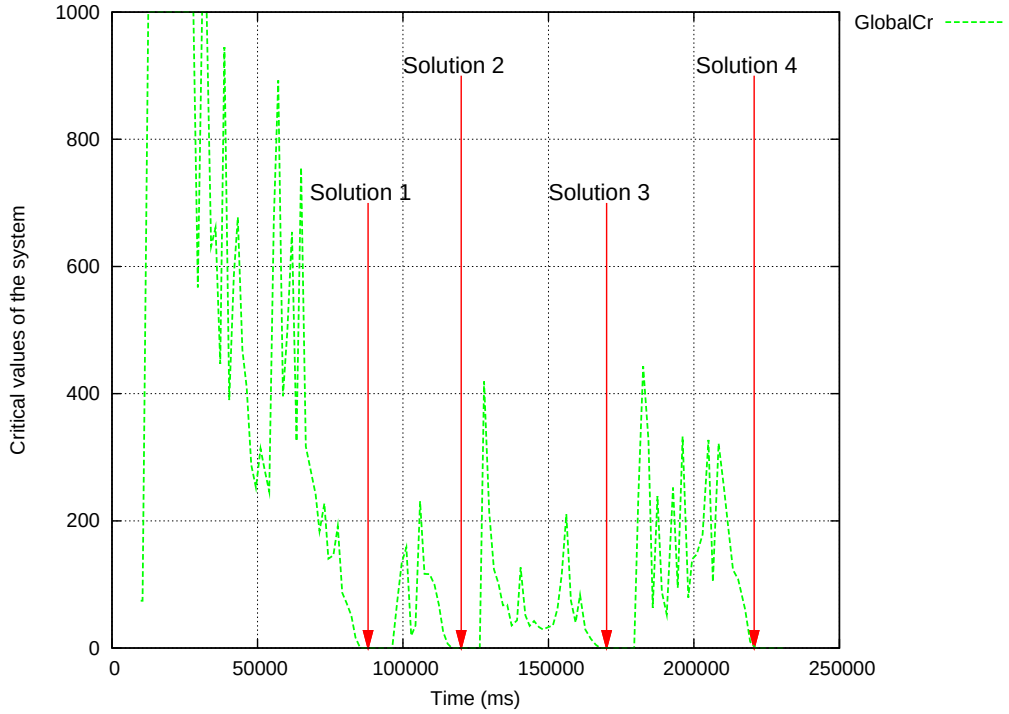


FIG. 8.4 – Évolution de la valeur critique du système

mètres et des performances varient fortement et la criticité évolue de manière discontinue. Cette discontinuité est due au besoin des agents d'apprendre progressivement les inter-dépendances du système en tenant compte des feedbacks environnementaux, tels que décrits en section 4.3.4. Finalement on peut dire que certaines valeurs changent fortement au début de la simulation puis oscillent autour d'une valeur avant de se stabiliser autour d'un compromis qui permet l'atteinte d'un état de convergence pour le système ; c'est par exemple le cas des paramètres $fuel$ et *span* (cf. figure 8.2).

8.1.2 L'adaptation du système aux interventions de l'utilisateur

MASCODE fournit des interfaces utilisateur qui aident à comprendre et gérer le système (cf. section 6.2.1). En utilisant ces capacités, le scénario précédent a été complété par l'introduction de nouvelles contraintes. Ainsi les indications de la figure 8.3 montrent que le système trouve trois autres solutions à la suite de changements introduits par le concepteur.

Les nouvelles contraintes changent l'équilibre du système en introduisant d'autres situations critiques (cf. figure 8.5). Un nouveau processus d'auto-régulation est alors automatiquement engagé, jusqu'à ce que le système seul trouve un nouvel équilibre.

Les figures 8.5 et 8.6 isolent les valeurs de quelques paramètres durant cette phase de

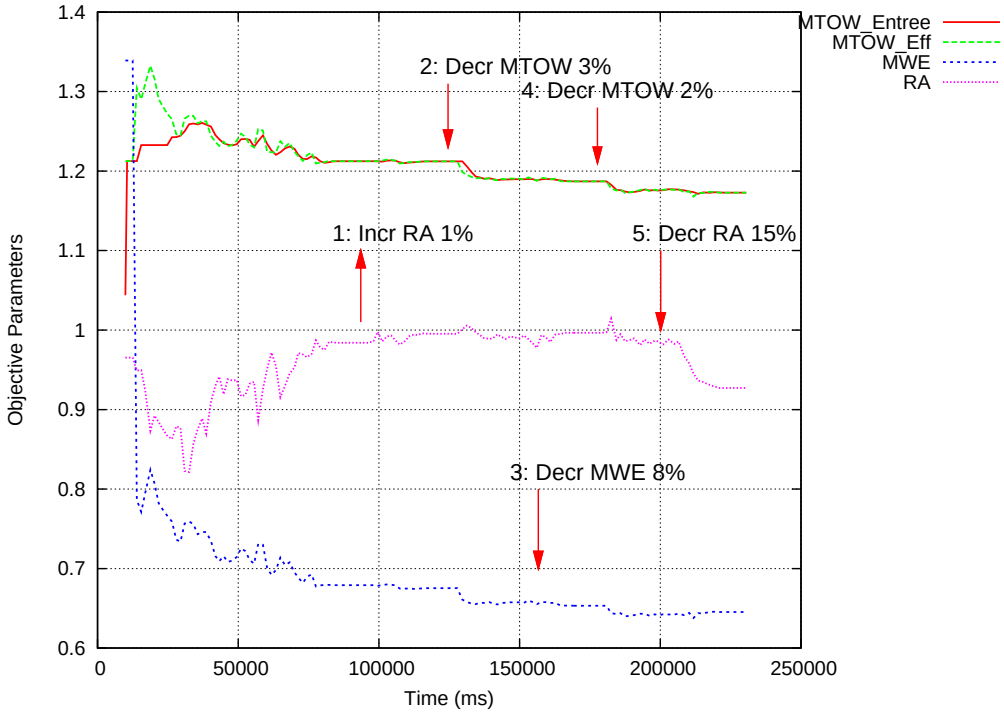


FIG. 8.5 – Valeurs des paramètres RA, $MTOW$ et MWE

résolution. Les paramètres RA (mission), $MTOW$ (masse au décollage) et MWE (masse à vide) sont des paramètres de haut niveau aux interdépendances fortes. Voici quelques exemples de relations qui les relient :

- la mission RA impacte directement la masse de carburant à emporter et donc la masse au décollage $MTOW$;
- augmenter la masse au décollage implique d'augmenter la masse à vide, car les contraintes structurales sont alors augmentées ;
- si la structure de l'avion change, alors la géométrie peut aussi évoluer ;
- si la géométrie de l'avion évolue, les forces aérodynamiques et la portance peuvent également être modifiées.

Pendant le processus de résolution des figures 8.5 et 8.6, des contraintes ont été changées par l'observateur de la manière suivante :

- Au temps $t = 84s$, l'objectif sur RA (mission) a été augmenté de 1%. Immédiatement une nouvelle criticité est associée, impliquant une modification sur $MTOW$ et MWE.
- À $t = 100s$, l'usager demande une diminution de la masse à vide $MTOW$. D'abord la valeur critique de $MTOW$ augmente, ensuite la contrainte est partagée entre RA, $MTOW$ et MWE. Cependant le système ne parvient pas à converger, car il est sur-contraint.

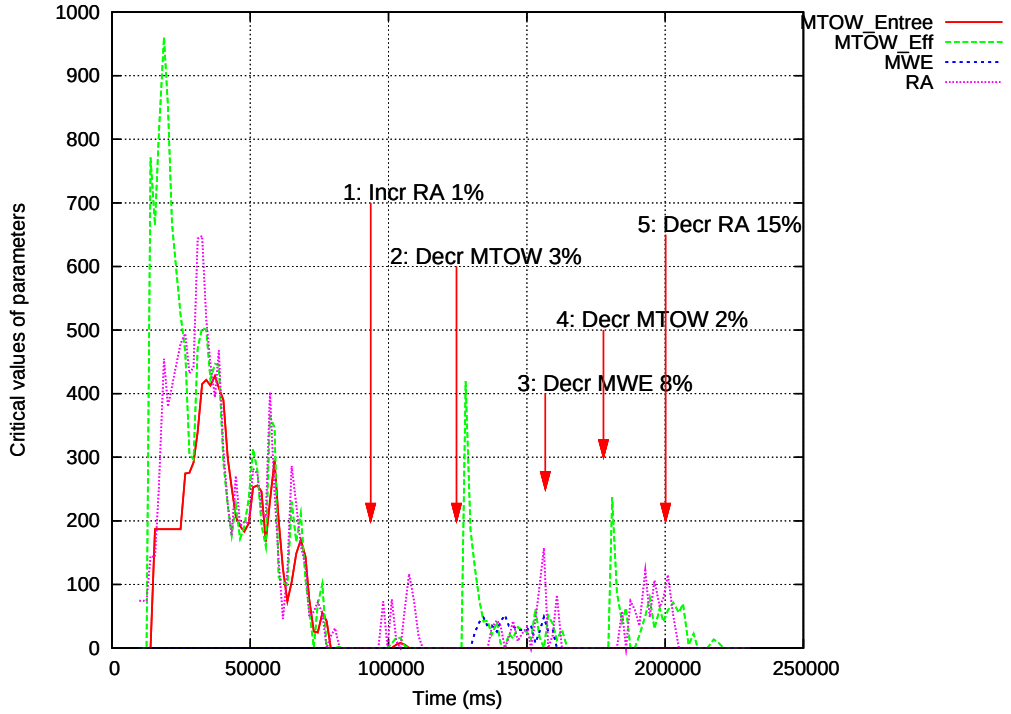


FIG. 8.6 – Valeurs critiques des paramètres RA, $MTOW$ et MWE

- À $t = 146s$, une modification de l'objectif MWE permet l'introduction de nouveaux degrés de liberté. En autorisant la diminution de la masse à vide, on permet au système de converger vers une solution, où la masse au décollage est faible $MTOW$ et le rayon d'action RA performant. Le système étant proche de la solution, on remarque qu'une faible modification de la contrainte MWE est suffisante pour l'obtention d'une nouvelle solution, figure 8.5.
- À $t = 150s$, la contrainte de $MTOW$ est renforcée (une baisse de 2%).
- À $t = 190s$, la performance sur la mission RA est dégradée, ce qui permet au système de converger, en raison des liens entre RA et $fuel$, et entre $fuel$ et $MTOW$.

Ce scénario illustre la capacité de MASCODE à prendre en compte et à s'adapter aux changements apportés par un utilisateur durant la phase de résolution. Ainsi chaque fois que la formulation du problème change, les agents adaptent leur comportement et recherchent un nouvel équilibre. En itérant des séries de modifications, l'observateur comprend les interdépendances et adapte sa conception en fonction de ses priorités. De plus lorsqu'un état stable n'est pas accessible[5], les agents auto-régulent alors les valeurs critiques parmi les parties conflictuelles et ainsi aident l'observateur à identifier les parties du système en contradictions. En utilisant les interfaces graphiques fournies par MASCODE, définie dans le chapitre 6, il identifie les conflits et modifie certaines contraintes

[5]le système est sur-contraint

	Test 1	**Test 2**
Agent	30	43
nb variables effectives	62	74
nb variables en entrée	15	17
nb variables en sortie	9	8
nb de bouclages	2	2
nb de critères contraints	49	59
nb de critères de sorties	5	5
nb de constantes	7	9

TAB. 8.1 – Caractéristiques de la comparaison

pour permettre au système de converger vers une solution.

8.1.3 Les effets de la distribution du système

Pour évaluer la sensibilité de l'approche aux caractéristiques du problème et la possibilité du passage à l'échelle, différents scénarios ont été testés, en faisant varier le nombre d'agents, le nombre de degrés de libertés ainsi que la dimension du problème.

L'augmentation du nombre d'agents

Pour évaluer l'impact du nombre des agents sur la résolution du problème, nous avons choisi de décomposer une des fonctions d'évaluation principale de notre modèle. La fonction choisie est la fonction réalisant l'évaluation de la masse (*weight*). Cette fonction se décompose en plusieurs sous-fonctions, permettant principalement d'évaluer la masse de chacun des composants de l'avion. Cette décomposition ne change donc pas vraiment la complexité du problème, puisqu'elle n'ajoute pas de nouvelles interdépendances fortes entres les objectifs principaux du problème. En revanche, elle modifie légèrement la manière de le résoudre, en ajoutant de nouveaux agents (voir tableau 8.8, et comparaison des figures 8.1 et 8.7).

En théorie, le nombre d'agents présents dans le système a un impact sur le nombre de messages échangés et sur les résolutions inverses effectuées. La figure 8.8 présente quelques résultats expérimentaux. Une première série de valeurs donne les temps de calcul nécessaires pour obtenir la convergence du système pour une approche à 30 agents (cas test 1) en utilisant respectivement 1, 2 et 3 machines. Une seconde série fournit les mêmes résultats mais pour une approche à 43 agents (cas test 2). Enfin la troisième série de résultats montre les temps de calcul que l'on aurait obtenus si l'ajout de 30% d'agents supplémentaires avait nécessité 30% de temps en plus pour converger (cas d'une complexité linéaire).

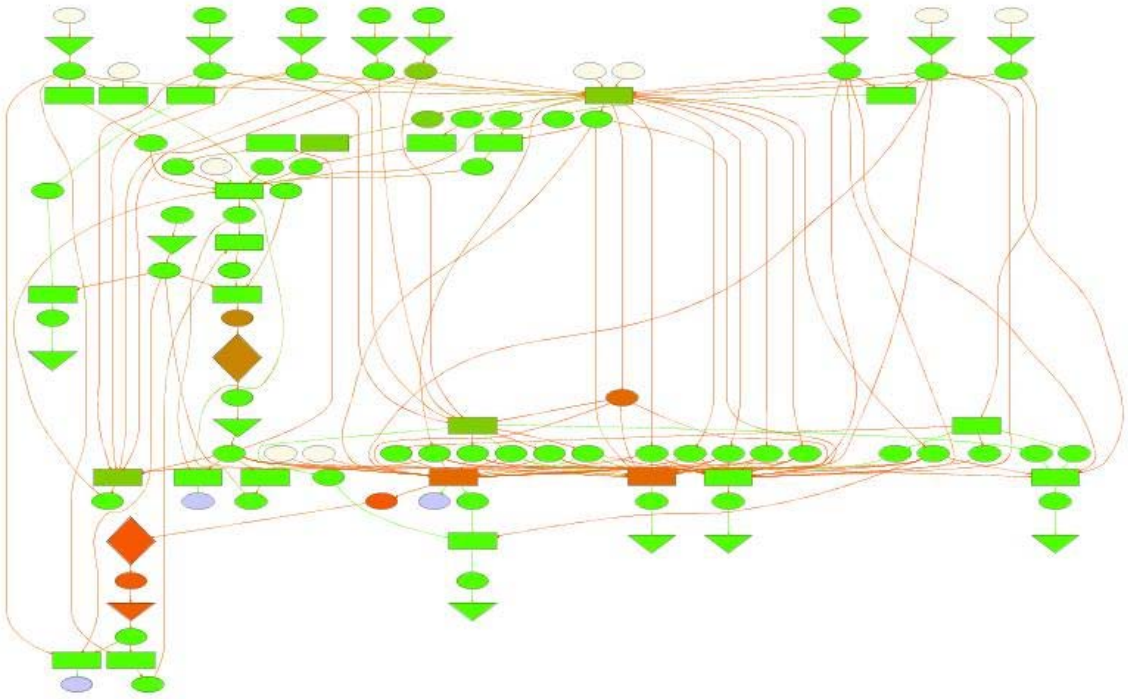


FIG. 8.7 – Décomposition du modèle de masse

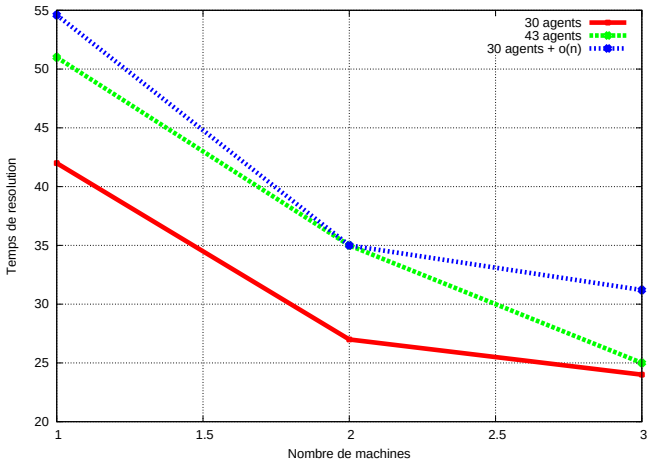


FIG. 8.8 – Impact du nombre d'agents sur le temps mis pour converger

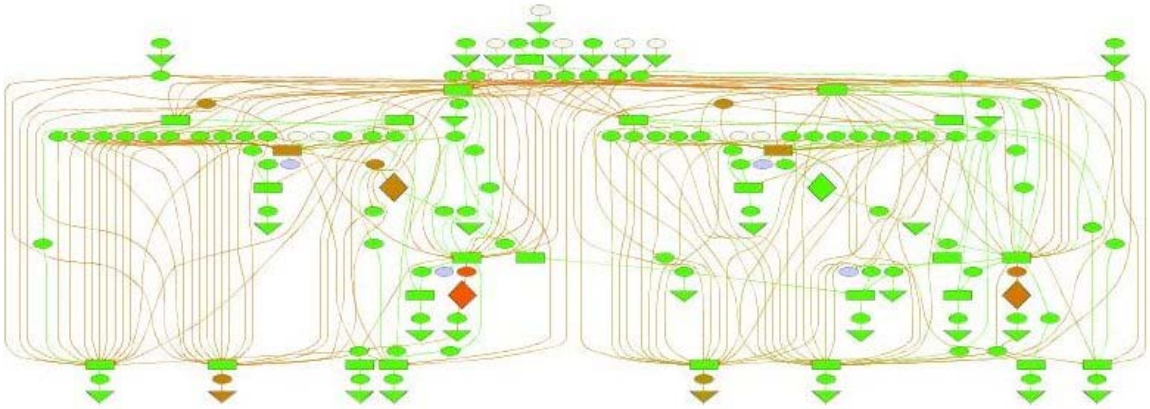


FIG. 8.9 – Cas tests incluant la conception de 2 avions

Ces résultats montrent que le temps mis par le système composé de plus d'agents, nécessite plus de temps pour converger, mais que cette différence reste limité et qu'elle devient négligeable dès que l'on distribue le problème sur 3 machines. D'autre part, les résultats ont aussi montré que le nombre d'itérations nécessaires aux agents pour résoudre ce problème est sensiblement le même selon que l'on utilise une approche à 30 ou à 43 agents, et qu'il se situe autour de 30 itérations. Ces résultats montrent donc que la résolution de notre problème n'est pas très sensible au nombre d'agents utilisé, lorsque la formulation du problème est la même.

L'influence de la taille du problème

Les résultats de l'expérimentation précédente ont permis de comparer l'impact du nombre d'agents sur la résolution d'un problème donné. D'autres expérimentations permettant d'envisager un passage à l'échelle sont nécessaires et réalisables en modifiant la formulation du problème.

Actuellement, la conception avion est amenée à considérer de plus en plus de contraintes. La construction de familles d'avions pose par exemple le problème suivant : peut-on trouver des avions qui partagent une voilure commune, tout en assurant des performances compétitives à l'ensemble des avions de la famille. Traiter ce type de problème revient à évaluer les configurations de plusieurs avions simultanément.

Nous avons défini un problème, pour lequel on cherche à concevoir deux avions qui partagent la même voilure, ceci se traduit par l'introduction d'une contrainte d'égalité sur les paramètres décrivant la voilure, ce paramètre restant par ailleurs un degré de liberté. Par rapport aux cas tests présentés précédemment, chaque modèle d'estimation d'une performance est dupliqué et associé à un des deux avions. Ainsi un nouveau système

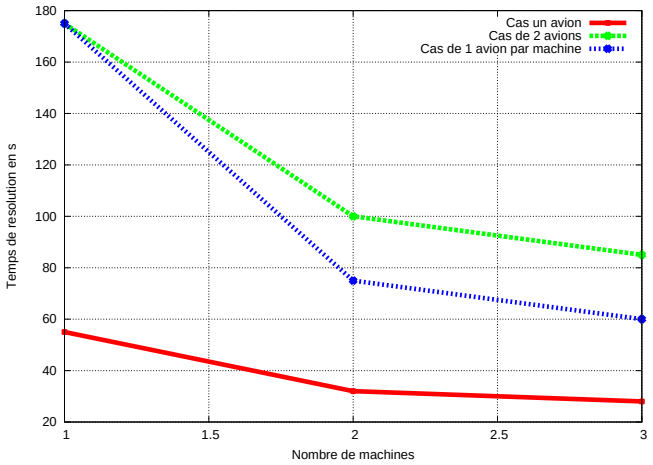


FIG. 8.10 – Influence de la taille du problème et de sa distribution

est assemblé tel qu'illustré sur la figure 8.9. Dans ce nouveau système, les deux avions partagent les paramètres définissant la voilure, représentés par des agents objectifs, qui ont à recueillir des demandes de modifications en provenance de chacune des deux grandes parties du système.

En passant à cette nouvelle formulation, le nombre de degrés de liberté a été légèrement augmenté passant de six à neuf, alors que le nombre de performances et de boucles a été doublé.

Les résultats des simulations sont présentés sur la figure 8.10. Ils montrent que les temps de calcul sont maîtrisés, puisque le passage à ce nouveau problème contenant plus de deux fois plus d'agents et deux fois plus de boucles de rétroactions, n'implique qu'une multiplication par 3 des temps de résolution.

L'influence du nombre de degrés de liberté

Les résultats précédents ont montré que finalement la complexité du problème est plus importante que le nombre d'agents lui-même dans les temps de résolution, puisque pour un même problème une augmentation du nombre d'agents de 30% implique un temps de

Nombres de degrés de liberté	**Élément fixé**
7	N/A
6	span
5	span tuc
4	fuel span tuc
3	phi fuel span tuc

TAB. 8.2 – Nombre de degrés de liberté et éléments fixés

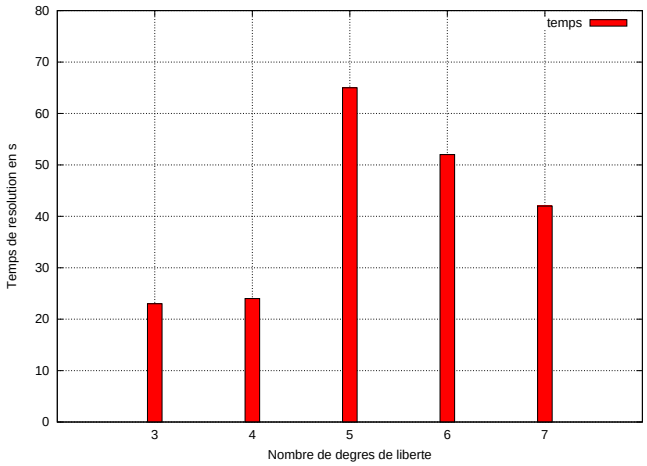


FIG. 8.11 – Influence du nombre de degrés de liberté

résolution inférieur à une augmentation de 30%, alors que la formulation d'un nouveau problème contenant deux fois plus d'agents et de boucles implique une multiplication par 3 des temps de calculs.

Un autre point étudié est l'influence du nombre de degrés de liberté pour trouver une solution. Nous nous plaçons ici dans le cas d'un problème mono-avion (figure 8.7), pour lequel on fait varier le nombre de degrés de liberté utilisés pour résoudre le problème. Le tableau 8.2 résume les différentes expérimentations menées. Ainsi à chaque fois que l'on retire un degré de liberté, on fixe la valeur du paramètre à une valeur solution obtenue précédemment. Les résultats de la figure 8.11 montrent qu'il n'est pas possible d'établir une relation entre la difficulté à converger et le nombre de degrés accordé au problème.

8.2 L'optimisation et le parcours du front de Pareto

8.2.1 La satisfaction de contraintes : comparatifs avec FSQP

Pour valider MASCODE, nous avons commencé par comparer nos résultats avec l'algorithme classique FSQP[78] (*Feasible Sequential Quadratic Programming*). Pour résoudre des problèmes inverses, FSQP se base sur un algorithme de gradient conjugué. Il est capable de faire de la satisfaction de contraintes puis de l'optimisation, et est connu comme étant rapide et relativement robuste. En utilisant le cas test présenté en section 8.1.1, nous avons comparé le temps mis par MASCODE pour trouver un point de convergence satisfaisant un ensemble de contraintes. MASCODE et FSQP cherchent des solutions en utilisant une amélioration de la solution obtenue. Le temps qu'ils mettent pour parvenir à un point solution dépend donc de leur capacité à améliorer les solutions. Ainsi, contrairement aux algorithmes génétiques ce temps n'est pas dépendant du nombre de paramètres

	FSQP	**MASCODE**
Temps de convergence	60*s*	70*s*
Hardware	P4, 3.2GHz dual core, 1Go RAM	P4, 3.2GHz dual core, 1Go RAM (peut être distribué : 20 threads)
Software	Scilab FSQP library	Plateforme Jade (Java) Scilab library

TAB. 8.3 – Comparaison temps de convergence pour MASCODE et FSQP

définissant le problème, mais uniquement des capacités de l'approche à se déplacer dans l'espace des solutions.

Les résultats de cette comparaison sont rapportés dans le tableau 8.3.

Les résultats montrent que les temps mis par FSQP et MASCODE sont comparables, mais légèrement meilleurs avec FSQP. Cependant et comme nous l'avons vu le temps mis par MASCODE dépend fortement du point de départ utilisé et de la distribution du système. Dans notre cas, le système est aussi beaucoup plus élaboré, il propose des interfaces graphiques des mémoires, etc. Ces résultats devront être complétés en comparant les nombres de cycles de résolution nécessaires tel que nous l'avons fait en section 8.1.3. Mais cette comparaison n'est pas facile, puisqu'une différence demeure entre MASCODE et FSQP, qui utilise un solveur de système pour résoudre le bouclage Masse-Performance. Ainsi pour pouvoir comparer pleinement les deux approches, il faudra trouver une nouvelle formulation pour FSQP qui permette d'intégrer l'égalité sur les boucles à l'aide d'une contrainte et non d'un solveur.
Pour conclure, ces résultats montrent que MASCODE trouve des solutions et ceci dans des temps de calcul plus que raisonnables en comparaison avec ceux de FSQP.

8.2.2 Le comparatif avec les algorithmes génétiques

La comparaison avec FSQP montre que MASCODE est capable de converger vers des solutions satisfaisant des contraintes. Mais FSQP n'est pas un algorithme traditionnellement utilisé pour la recherche de front de Pareto. Pour comparer les résultats d'optimisation de notre approche, nous avons donc utilisé un algorithme évolutionnaire et mémorisé les solutions de Pareto en utilisant une approche semblable à celle d'un algorithme de PESA [20] (cf. section 5.1.1).

	Algorithme génétique	MASCODE
Degrés de liberté	Awing, Fnslst	idem + MTOW, RA
Contraintes	Kfn, Vz, Fff, OWE, PL	idem
Objectifs	Vapp, tofl, MTOW, RA	idem

TAB. 8.4 – Comparaison des formulations MASCODE et AG

La formulation du problème pour l'AG

Pour simplifier la visualisation des résultats et permettre à notre algorithme génétique de trouver facilement des solutions au problème, nous avons réduit le nombre de degrés de liberté à deux (la poussée *Fnslst* et l'envergure *Awing*), mais conservé les contraintes sur les performances et défini quatre objectifs sur : la vitesse approche (*Vapp*), la masse au décollage (*MTOW*), le rayon d'action (*RA*) et la distance au décollage (*tofl*). Cette formulation de problème est assez intéressante, car les quatre objectifs choisis ont des demandes contradictoires sur nos deux degrés de libertés comme nous le verrons en comparant les résultats.
La différence majeure entre MASCODE et l'Algorithme Génétique (AG) réside dans le fait que MASCODE cherche des solutions aux paramètres de conception et aux boucles de rétroactions simultanément alors que l'AG utilise une résolution de systèmes et procède donc en suivant les deux étapes :

1. définir les valeurs du génotype d'un individu, dans notre cas choisir une valeur pour *Fnslst* et *Awing* ;
2. utiliser un analyseur du système permettant de trouver des valeurs pour les boucles, qui soient cohérentes.

Durant la première étape, il s'agit simplement de créer un nouvel individu en utilisant l'algorithme génétique. Dans un second temps, on cherche les valeurs des boucles qui permettent de converger vers une solution homogène. Cette décomposition en deux étapes bien distinctes est de type MDF (*Multi-Disciplinary Feasible*, cf. section 2.2.2).

À l'inverse dans MASCODE, l'adaptation se fait simultanément au niveau des boucles et des degrés de libertés. Ainsi à chaque itération, les agents boucles de rétroactions modifient leurs valeurs de paramètre (*MTOW* et *RA*) en même temps que les agents objectifs d'entrées modifient les leurs (*Awing* et *Fnslst*). Ces deux formulations de problème sont comparées dans le tableau 8.4.

Les résultats de la comparaison

Les résultats ont été obtenus en mémorisant les solutions non-dominées trouvées par chacune des approches. Au niveau de l'algorithme génétique noté AG I, ceci se traduit

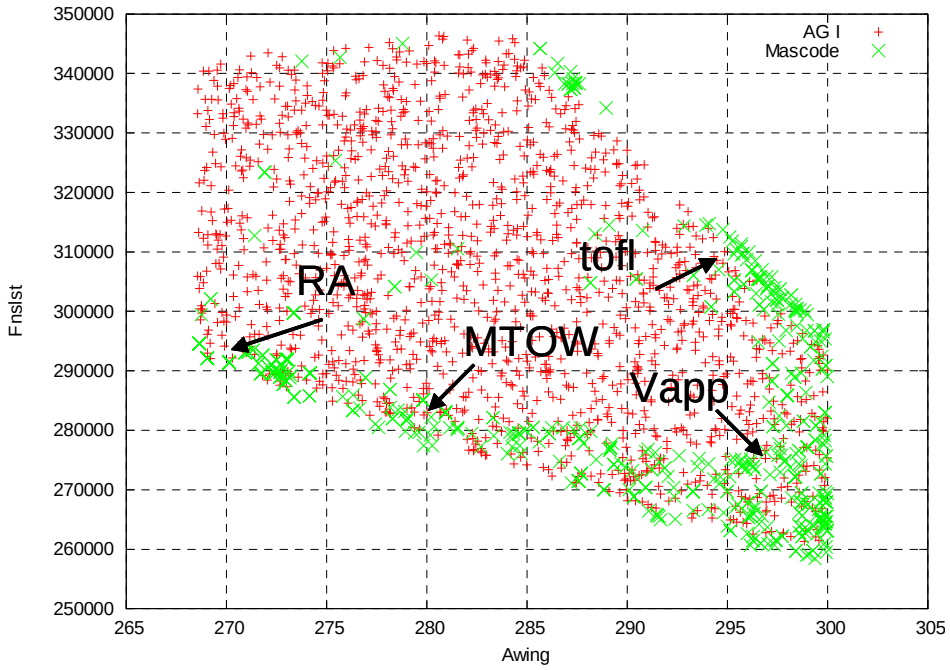
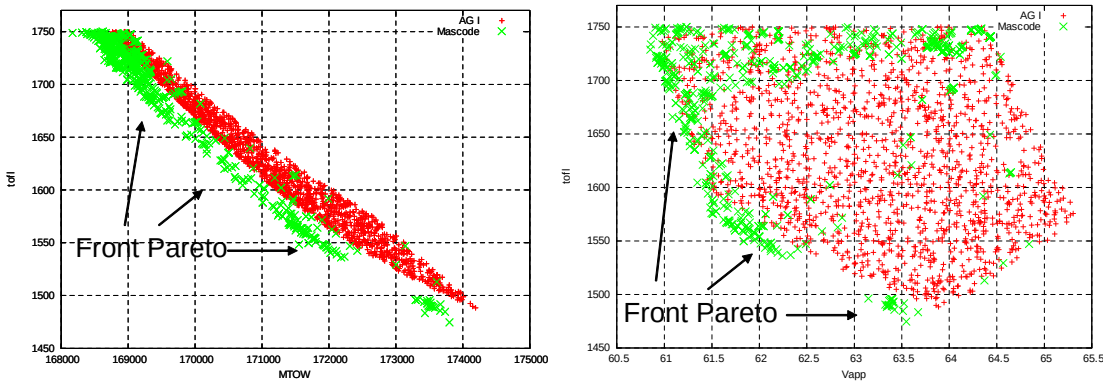


FIG. 8.12 – Répartition des solutions non-dominées sur les degrés de liberté (*Awing*, *Fnslst*)

par la construction d'une population d'individus non-dominés, et pour MASCODE, par l'utilisation de l'algorithme de modifications de contraintes de l'agent architecte (cf. section 5.4).

Les figures 8.12 et 8.13 comparent les solutions non-dominées trouvées par chacune des deux approches. La figure 8.12 trace l'espace des solutions non-dominées trouvé pour *Awing* et *Fnslst*, et la figure 8.13 des fronts de Pareto pour les objectifs *MTOW*, *tofl* et *Vapp*. On remarque sur la figure 8.12 que MASCODE trouve principalement des solutions aux frontières de l'espace défini par *Awing* et *Fnslst*. En fait, les optimisations des différents objectifs (*Vapp*, *MTOW*, *RA* et *tofl*) ont tendance à attirer les solutions vers les frontières, tel que l'illustre les flèches de la figure 8.12. Or, notre manière d'optimiser les solutions commence toujours par optimiser un des critères puis à le dégrader pour en optimiser d'autres (cf. section 5.4). Pour obtenir un ensemble de solutions comparables à l'AG I, il suffirait donc de s'autoriser des dégradations de nos critères plus importantes afin de converger vers des solutions de plus grand compromis. Cependant en augmentant le nombre de points à rechercher on augmente la combinatoire, or ce n'est pas forcément en adéquation avec notre objectif qui est de trouver les frontières de l'espace de recherche rapidement et de proposer des fronts de Pareto entre les objectifs. Or au vu des figures 8.13(a) et 8.13(b), on peut dire que notre approche répond assez bien à cette exigence. Sur ces figures, on s'aperçoit que MASCODE est capable de trouver de meilleures



(a) Objectifs *MTOW-tofl* (b) Objectifs *Vapp-tofl*

FIG. 8.13 – Fronts de Pareto

solutions aux fronts de Pareto, ce que nous expliquons par les trois hypothèses suivantes :

1. En commençant par chercher la solution optimale pour chaque objectif, on converge vers une solution non-dominée sans être dépendant du paramétrage de notre algorithme, tel qu'illustré en section 5.4.2 (MASCODE est donc robuste et capable de trouver de meilleures solutions).
2. L'analyseur de système utilisé par l'AG I est simple, il obtient les valeurs des paramètres $MTOW$ et RA en utilisant une itération de type FPI (*Fixed Point Iterative*), cf. section 2.2.1, qui fait que les valeurs obtenues pour $MTOW$ et RA après convergence peuvent ne pas être optimales.
3. Pour permettre à MASCODE de converger, un écart limite entre les valeurs des paramètres de boucle ($MTOW$ et RA) est toléré. Par exemple, une différence entre la valeur de $MTOW$ fournie au système et celle effectivement calculée est acceptée et doit être inférieure à 0.2% de la largeur de l'intervalle du paramètre. En acceptant un écart de 0.2% pour les paramètres $MTOW$ et RA, on introduit quelques incertitudes susceptibles de décaler légèrement les fronts de Pareto. Il sera donc souhaitable de mesurer ce phénomène en relation avec la thèse sur la propagation des incertitudes lors de la conception avion, qui débutera à EADS IW dans le courant 2008.

Les limites des algorithmes génétiques

Pour tenter de répondre aux trois hypothèses précédentes, nous avons changé la formulation de l'algorithme génétique en supprimant le solveur de système et en définissant des contraintes d'égalité sur les paramètres $MTOW$ et RA qui sont alors identiques à celles de MASCODE. Ceci nous a permis d'obtenir une nouvelle formulation de type AAO (*All At Once*, cf. section 2.2.2), que nous notons AG II et d'obtenir une série de résultats que nous

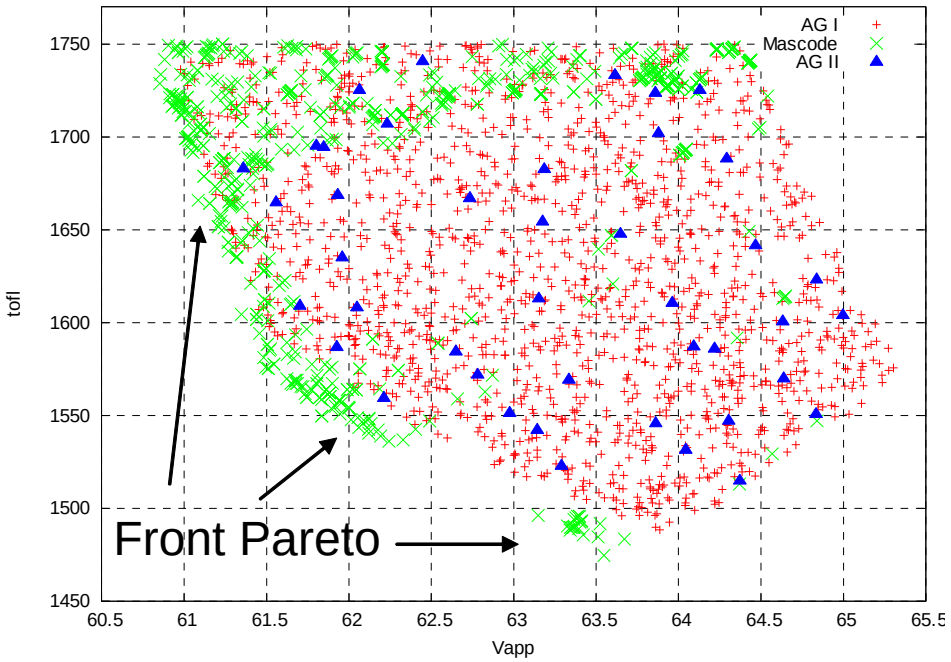
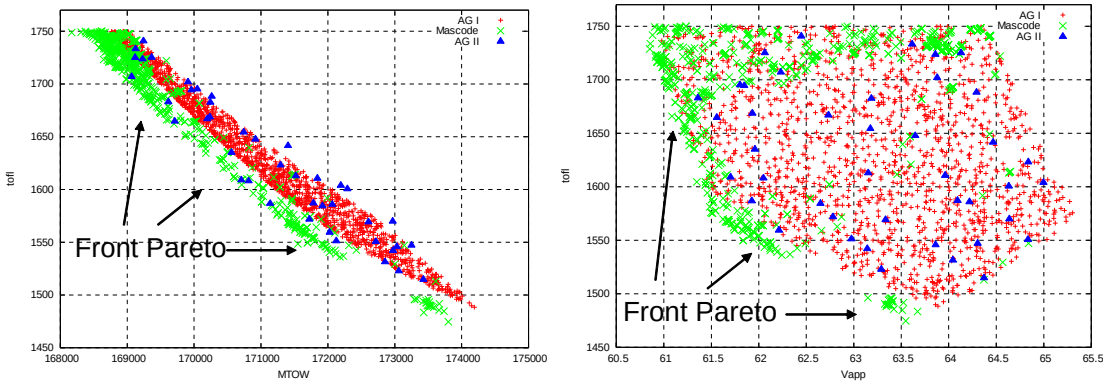


FIG. 8.14 – Répartition des solutions non-dominées pour les 3 approches



(a) Objectifs *MTOW-tofl* (b) Objectifs *Vapp-tofl*

FIG. 8.15 – Fronts de Pareto

comparons aux deux précédentes sur les figures 8.14 et 8.15. Ces résultats montrent qu'il devient difficile pour l'AG II de trouver un ensemble de solutions non-dominées, certainement à cause de l'augmentation de la combinatoire due aux nouveaux degrés de liberté que sont *MTOW* et *RA*. En revanche, de nouvelles solutions comparables à celles de MASCODE et dominantes par rapport à celle de l'AG I sont trouvées, cf. figure 8.15(a). Mais des solutions trouvées par MASCODE sont toujours dominantes (figures 8.15(a) et 8.15(b)), ce qui nous conforte dans nos hypothèses précédentes :

- MASCODE trouve assez facilement des solutions non-dominées ;
- nos contraintes d'égalité introduisent un biais qui décale légèrement les solutions de Pareto ;
- l'analyseur à base de FPI de l'AG I empêche sa convergence vers les meilleures solutions.

Nous ne pouvons malheureusement pas conclure sur l'implication précise de chacun de ces derniers points. Mais le travail sur la propagation d'incertitudes qui doit débuter l'année prochaine et en liaison avec notre problème, permettra certainement d'apporter des éléments de réponse.

Bilan de la comparaison

Cette comparaison avec les algorithmes génétiques, nous montre que MASCODE est capable de trouver de très bonnes solutions de Pareto à un problème de conception avion multi-objectif et multidisciplinaire sans avoir à ajuster son paramétrage. Ceci nous permet de valider notre modèle de satisfaction de contraintes par auto-organisation, ainsi que notre approche pour parcourir des solutions de Pareto à l'aide d'un agent de niveau supérieur (architecte).

D'autre part, il est important de noter que la formulation du problème que nous avons traitée ici a été choisie pour que l'algorithme génétique n'ait aucune difficulté à trouver des solutions au problème, puisque notre objectif n'était pas de fournir un algorithme génétique optimal mais bien de comparer nos résultats. De ce fait, nous avons utilisé un problème à deux degrés de liberté pour limiter la combinatoire de l'AG, ce qui est un avantage fort pour cette méthode. En particulier, le changement de paramétrage de l'AG II avec l'ajout de contraintes d'égalité sur *MTOW* et *RA*, a montré sa sensibilité à une augmentation du nombre de degrés de libertés, rappelant ainsi la robustesse de notre approche sur ces aspects (cf. section 8.1.3).

Ainsi cette comparaison nous permet de conclure que MASCODE est :

- capable de trouver de très bonnes solutions de Pareto, sans avoir à modifier son paramétrage ;
- insensible à la combinatoire engendrée par le nombre de degrés de liberté, ce qui

permet de traiter des problèmes comportant un nombre de degrés de libertés important.

8.3 Synthèse des capacités de MASCODE

Les différentes expérimentations que nous avons présentées dans ce chapitre, nous permettent de tirer les conclusions suivantes :

- MASCODE est capable de converger vers un état d'équilibre :
  1. lorsqu'une solution au problème est trouvée, la criticité des agents est nulle ;
  2. lorsque le système est sur-contraint la criticité globale est non nulle, mais répartie entre les agents en conflits.
- MASCODE est robuste aux modifications introduites par un concepteur. Ainsi si les contraintes du problème sont changées, les agents trouvent un nouvel état d'équilibre en propageant les nouvelles contraintes d'une manière fortement discriminante (seules les parties du système concernées sont impactées) ;
- la capacité de MASCODE à converger rapidement vers des solutions n'est pas dépendante d'une combinatoire due au nombre de degrés de liberté. Ainsi la difficulté à trouver des solutions est indépendante du nombre de degrés de liberté. En revanche, le point d'initialisation de la résolution et le nombre de situations conflictuelles rencontrées au cours de la résolution qu'il implique, sont des facteurs de complexité pour MASCODE. Ceci explique que lorsque nous avons joué sur le nombre de degrés de liberté, nous avons eu parfois des temps de résolution plus importants sur le même problème pour un nombre de degrés de liberté plus faible ;
- la taille du problème n'est pas non plus un facteur de complexité déterminant, puisqu'une augmentation du nombre d'agents implique une augmentation linéaire du temps de résolution.
- MASCODE est capable d'optimiser des solutions et de construire des fronts de Pareto grâce à ses capacités d'adaptation aux modifications et à la stratégie de son agent architecte.

Tous ces aspects font de MASCODE une approche prometteuse pour une utilisation industrielle. Mais l'introduction de plus de degrés de liberté implique aussi une nécessité forte d'explication et de comparaison des solutions pour ne pas déstabiliser l'utilisateur par une quantité d'information trop importante et inhabituelle. Ainsi pour proposer une approche compréhensible et complètement interactive, il faudra certainement voir comment on peut interagir régulièrement avec lui et intégrer certains de ses choix dans le raisonnement de notre agent architecte.

9
Les perspectives

Sommaire

Les perspectives à MASCODE peuvent être décomposées en plusieurs axes :

- améliorer les mécanismes de coopération en considérant d'autres facteurs d'auto-organisation, que ceux de la régulation ;
- améliorer l'heuristique permettant le parcours du Front de Pareto en donnant plus de capacités de raisonnement à l'agent architecte ;
- considérer des scénarios utilisateur afin d'approfondir les méthodes d'interaction et la co-construction des solutions ;
- confronter l'approche à d'autres cas d'applications pour en mesurer les limites et tendre vers une approche générique.

9.1 Les besoins d'autres mécanismes d'auto-*

En utilisant un comportement auto-régulateur, MASCODE permet de prendre en compte une partie des aspects multidisciplinaires et multi-objectifs de la conception. Cependant selon E. Morin [59], un système auto-organisé complet exige d'autres capacités

telles que l'auto-liaison et l'auto-maintenance. La capacité d'auto-liaison permet au système de construire des organisations à partir de comportements locaux, alors que l'auto-maintenance sert à évaluer la pertinence de l'organisation et à la réorganiser si nécessaire.

9.1.1 L'auto-organisation des modèles

La complexité du problème dépasse l'optimisation simple des paramètres objectifs, car le choix des modèles impliqués dans la fonction globale, est également un facteur important sur la performance de la résolution. En effet, la fonction globale à évaluer est différente pour chaque étude, puisqu'elle est composée de modèles calibrés et ajustés individuellement en fonction du contexte puis assemblés pour former la fonction globale (section 1.2.5). Par la suite, le comportement global du système est dépendant de la modélisation de chacune de ses parties et donc du calibrage individuel des modèles. En fonction du comportement global de la fonction, un concepteur modifie ses objectifs globaux, ce qui a aussi un effet sur les modèles locaux. De ce fait, les modifications de contraintes, les changements d'hypothèse sur les modèles, etc. participent à la recherche de meilleures solutions. Favoriser la co-existence de plusieurs modèles de simulation et permettre aux agents de choisir le modèle le plus adapté en fonction de l'évolution du contexte et des besoins des autres disciplines, est donc également un facteur d'amélioration de la conception avion et de la qualité/robustesse/précision des solutions obtenues.

Pour résumer, en modifiant les paramètres objectifs, les agents font évoluer le contexte de la résolution. Ces évolutions peuvent être de différents ordres, et impliquer différents types de réorganisation du système, répondant à des besoins :

de recalibrage ou de changement de modèle : ces changements génèrent à leur tour une évolution du contexte dans d'autres parties de la résolution, ce qui nécessite un recalibrage ou un changement d'autres modèles. Dans ce genre d'approche, la fonction globale est donc adaptée et réorganisée en permanence en fonction des interactions entre les agents ;

de gestion des niveaux de détail des modèles : supposons que des agents en utilisant des modèles peu détaillés, trouvent un premier consensus sur des valeurs de paramètres objectifs, on pourrait alors décider de préciser la solution en utilisant des modèles plus fins (de granularité différente). De cette manière, on prend en compte plus d'aspects et on précise la solution. Mais les résultats obtenus avec ces modèles plus fins pourraient aussi servir à recalibrer les modèles moins détaillés, en procédant ainsi étape par étape.

À l'inverse supposons que les agents utilisant des modèles détaillés ne parviennent pas à trouver un consensus, ils pourraient alors faire appel à des modèles moins fins

afin d'évaluer des tendances et de guider la résolution.

Afin de tirer parti de ces phénomènes de réorganisation, il est nécessaire de trouver de nouveaux critères de coopération, qui permettent de faire évoluer l'organisation du système sans qu'elle ne devienne un désordre permanent. La définition de ces mécanismes de réorganisation est donc une étape importante pour des travaux futurs. Mais son étude ne pourra vraiment commencer qu'après avoir pu :

- identifier et formaliser des connaissances sur les modèles pour pouvoir raisonner, telles que la granularité, la précision, le temps de calcul, etc. ;
- permettre l'intégration, la co-existence de modèles très hétérogènes au sein d'une même application.

9.1.2 L'amélioration de la coordination des agents

Le choix d'un modèle de coordination pour la simulation/résolution d'un système complexe est important, car il a des répercussions sur les connaissances et les modèles de décision qu'il faut donner aux agents pour obtenir un comportement collectif cohérent.

Dans notre approche nous avons choisi un modèle de coordination en utilisant la structure du problème afin de synchroniser les envois et les réceptions de messages, puisque chaque agent attend de recevoir soit l'ensemble de ces messages d'exécution, soit l'ensemble de ses messages de modification avant d'agir. Mais une autre approche plus autonome et plus auto-organisatrice est possible. Elle consiste à laisser le choix de l'interaction (envoi de message) à l'initiative de l'agent en lui permettant ainsi d'adapter l'envoi des messages à ses besoins de négociations. Par conséquent, ceci présuppose que de nouveaux comportements coopératifs soient définis notamment par l'identification de nouvelles règles de non coopération, sans lesquelles le contrôle de l'envoi des messages ne peut fonctionner correctement, ce qui conduirait, par exemple :

- **à l'augmentation des informations transitant dans le système**, puisque chaque message reçu par un agent pourrait potentiellement en générer N autres en collaborant avec chacun de ses voisins ;
- **à une saturation de certains agents**, puisque les inégalités de sollicitation et la topologie du système peuvent conduire rapidement des goulots d'étranglement.

En considérant la charge locale d'un agent, nous avons défini de nouvelles règles de régulation, qui ont permis de contrôler les envois de messages ainsi que les exécutions.

Par exemple, lorsqu'un agent reçoit beaucoup de demandes, il ne traite que les demandes les plus critiques et dont la variation est supérieure à son pas de modification minimum. Lorsque le nombre de demandes diminue au cours de la résolution, l'agent diminue son pas de modification minimum, ce qui lui permet de traiter toutes les demandes

reçues en fin de simulation. Nous avons fait le choix d'adapter le comportement de l'individu à sa charge de travail. Des expérimentations, nous ont permis de converger mais en utilisant des ressources et des temps de résolution plus importants. Par la suite nous avons orienté nos travaux vers l'optimisation des solutions et les interactions avec l'utilisateur laissant ces aspects de côté.

Mais il serait intéressant d'approfondir et d'explorer ces aspects, en utilisant une attitude plus coopérative, qui consisterait à adapter le comportement de l'individu non pas uniquement par rapport à sa charge de travail mais aussi par rapport à celle de son voisinage. Cela se traduirait par la règle suivante : un agent envoie des informations à un voisin seulement si celui-ci est capable de les traiter. Ainsi les règles de coopération à mettre en place devraient adapter l'envoi de message à la capacité des autres à les traiter. Un lien avec les travaux de Parunak et al. sur l'hyperactivité des agents serait également à considérer [63].

9.1.3 L'amélioration du parcours du front de Pareto

Comme nous l'avons vu, l'optimisation est nécessaire et importante, elle permet de comparer des solutions en parcourant l'espace de recherche. La stratégie que nous avons proposée présente une certaine généricité. Cependant, lorsque le nombre de degrés de liberté devient important et que les objectifs visés sont multiples, il devient difficile à la fois de parcourir l'espace des possibilités et de présenter l'information de manière à faciliter la prise de décision. Or l'interaction avec l'utilisateur sur les caractéristiques d'un problème de conception repose sur sa compréhension des interdépendances entre les objectifs (cf. section 2.1.4).

Ainsi, l'amélioration de l'optimisation des solutions doit se faire selon deux aspects :

améliorer les interactions utilisateur : en offrant d'autres outils de visualisation, nous pensons que notre approche peut apporter d'autres moyens d'explications et d'interactions, qui puissent améliorer la compréhension du système et favoriser l'optimisation des solutions, sans augmenter la charge cognitive du concepteur ;

améliorer la stratégie de parcours : la sélection des sous-objectifs à optimiser permet de parcourir plus ou moins efficacement l'espace de recherche. Ainsi, nous pensons pouvoir améliorer son efficacité en considérant et en apprenant les interdépendances locales entre les sous-objectifs à optimiser. Cette stratégie consiste à optimiser en premier les critères, qui sont les moins contraints et les moins interdépendants, et donc ceux pour qui l'optimisation a le plus de chance de réussir.

9.2 L'amélioration de l'interaction avec le(s) concepteur(s)

Comme nous venons de l'évoquer, nous préconisons d'utiliser des approches décisionnelles interactives, qui permettent d'orienter le parcours de l'espace de recherche aux solutions intéressantes, et qui aident un concepteur à faire évoluer son problème en l'informant, par exemple, des parties du système qui sont conflictuelles, des modifications de contraintes qui lui permettent d'améliorer sa conception, des objectifs qui sur-contraignent le problème, etc.

Pour répondre à ces objectifs, on peut construire des scénarios à partir des propositions faites dans le chapitre 6 :

- suivi de la résolution à l'aide d'indicateur ;
- identification des demandes de modifications conflictuelles ;
- modification des contraintes de certains objectifs.

L'objectif de ces scénarios serait d'identifier à partir de nos propositions des moyens de présenter les informations pertinentes et de proposer des modifications possibles de manière à renforcer la coopération avec le concepteur.

Ces scénarios aboutiront, par exemple, à l'identification automatique des paramètres pour lesquels les contraintes et interdépendances sont fortes, à des propositions de relaxation de contraintes ou à des propositions d'optimisation de certains critères.

9.3 Vers un atelier générique d'aide à la conception de produits complexes

Durant cette thèse nous avons vu la capacité des agents à traiter des problèmes multidisciplinaires, multi-objectifs, pour lesquelles les interdépendances sont nombreuses et les connaissances hétérogènes et distribuées au sein d'un collectif d'agents. Au-delà de la conception préliminaire avion, on retrouve ce type de problématique à toutes les étapes de la conception de produits complexes (avions, sous-marins, satellites, etc.). Ainsi une généralisation de notre architecture devrait permettre la résolution d'autres problèmes de conception notamment en ingénierie des systèmes pour lesquels les connaissances et objectifs à prendre en compte sont nombreux, hétérogènes, distribués et interdépendants.

L'application de nos travaux de thèse à d'autres cas d'application est donc importante, car elle permettrait d'en identifier les limites actuelles et d'en améliorer la généricité.

L'objectif à long terme est donc de parvenir au développement d'un atelier d'aide à la conception, qui permettrait à différents concepteurs (métiers) de définir leurs objectifs

et les degrés de libertés, et de les lier à l'aide de fonctions métiers permettant leur évaluation croisée. Ensuite des contraintes de conception dues à des objectifs plus globaux (réduction de coûts) et des relations avec d'autres métiers (besoins électriques, résistance des matériaux) seraient ajoutées de manière à proposer une modélisation complète d'un problème et à favoriser la prise en compte d'un maximum de contraintes. De cette manière chaque concepteur améliorerait la compréhension de son problème et favoriserait la remonté d'informations à un plus haut niveau (architecte), afin de rechercher des compromis avec d'autres parties/systèmes également en cours de conception.

Conclusion générale

Après avoir analysé les méthodes d'aide à la conception préliminaire avion, nous avons vu que le cadre théorique de ces approches est basé sur la réduction du problème avec pour objectif sa transformation en une formulation mathématique. Mais concentré sur la transformation du problème en sous-problèmes et sur sa réduction paramétrique, ces approches finissent par aboutir à une limitation des possibilités de conception et à une mauvaise compréhension globale du problème alors considéré comme une boîte noire. L'introduction d'incertitudes dans les outils de conception a aujourd'hui pour but de qualifier les résultats en y ajoutant de l'information. Cette voie intéressante, apporte une nouvelle façon d'aborder les problèmes en favorisant la compréhension et la recherche de solutions, qualifiées robustes et moins risquées.

Ces nouvelles approches permettent certes de mieux appréhender le comportement local et global de ces boîtes noires, en qualifiant les résultats. Mais ces méthodes sont insuffisantes, puisque la conception est un processus vivant qui nécessite le déploiement d'outils dynamiques, évolutifs, ouverts, distribués, finalement proches des organisations humaines et industrielles dans lesquelles les produits sont conçus. En nous rapprochant d'un cadre scientifique lié à celui de la modélisation des systèmes complexes, on s'autorise de nouveaux degrés de liberté et une nouvelle manière non-réductionniste d'aborder ces problèmes. Ce choix est aujourd'hui possible grâce à l'émergence progressive de nouveaux outils informatiques issus de la vie artificielle, qui sont riches au niveau de la sémantique, du raisonnement et surtout capables d'interagir et d'apprendre collectivement. L'utilisation de ces capacités permet aux éléments de s'auto-organiser et favorise ainsi l'émergence de nouveaux compromis de conception intégrant un nombre important de données, de connaissances, de choix et d'interactions.

Tout au long de ce travail, nous avons illustré ces capacités en déployant un système multi-agent, en l'appliquant à des problèmes tests, et en comparant les résultats à une méthode classique d'optimisation. Ainsi, nous avons montré qu'il est possible d'utiliser ces méthodes pour aider à la résolution de problèmes de conception difficiles, que les résultats fournis sont cohérents et comparables avec ceux de l'état de l'art, et surtout, nous avons illustré les perspectives qu'elles permettent d'envisager en terme de compréhension, de modélisation, d'intégration de données et de connaissances diverses et de négociation. Tous ces aspects participent à la résolution de problèmes complexes pour lesquels l'humain est incapable de maîtriser toutes les connaissances et toute la dynamique.

Cependant le chemin pour parvenir à une méthode générique facilement applicable à de nombreux problèmes de conception avion est encore long, il passera certainement par

le développement des perspectives suivantes :

- définir une méthode de modélisation agent simplifiée, qui soit utilisable par des ingénieurs en ingénierie des systèmes. Cette méthode aboutirait à la définition d'exigences ;
- proposer ou trouver des outils de modélisation fonctionnelle, qui permettent la description des points de vue simplifiés ;
- travailler sur des interfaces utilisateur pour améliorer la compréhension et la redéfinition de problèmes ;
- améliorer le suivi du système et la vue architecte ;
- enrichir les capacités des agents en leur fournissant plus d'autonomie sur les changements de modèles.

Les apports à la conception

Un prototype prenant en compte les problématiques métiers de la conception préliminaire avion a été développé en utilisant un SMA adaptatif. Cette réalisation a donné lieu à un outil d'aide à la conception, qui apporte de nouvelles solutions aux concepteurs en prenant en compte d'avantage de dimensions du problème.

Ainsi MASCODE offre des propriétés d'adaptation aux changements, de robustesse à l'augmentation de la dimension d'un problème, d'interaction, de résolution de conflits, etc. Comme nous l'avons vu, ces résultats ont été obtenus en utilisant les propriétés des agents coopératifs, dont le comportement collectif émerge des actions individuelles.

Aujourd'hui le démonstrateur MASCODE propose des interfaces graphiques d'aide au pilotage du système. Elles permettent l'intervention d'un utilisateur et le suivi de la résolution. Comme nous l'avons décrit précédemment, ces interfaces devront être complétées et réutilisées dans des scénarios de conception afin d'aider les concepteurs à s'approprier la technologie et de mieux cibler leurs besoins. D'autre part en traitant de nouveaux problèmes, on devrait pouvoir identifier les nouvelles spécificités de MASCODE et trouver des moyens pour l'enrichir et l'appliquer à d'autres problèmes.

Les apports à la théorie des AMAS

En appliquant la théorie des AMAS à un problème industriel, nous avons proposé un algorithme de résolution de contraintes pour des problèmes continus multidisciplinaires et multi-objectifs. Cette expérience nous a permis d'identifier les avantages conceptuels, qu'offrent les SMA adaptatifs et de proposer une approche originale, car à l'intersection de deux domaines bien différents que sont l'optimisation multi-objectif et multidisciplinaire, et l'optimisation combinatoire à base d'agents.

Concernant la théorie des AMAS, il n'existe aucune preuve mathématique permettant de garantir leur efficacité à résoudre des problèmes. En identifiant les situations non coopératives, et en définissant des mécanismes de coordination et de communication associés au raisonnement coopératif, nous avons montré la capacité des AMAS à résoudre ce type de problème et comparé les résultats à une approche à base d'algorithmes génétiques. Les résultats obtenus renforcent l'intérêt des AMAS par rapport à d'autres algorithmes, et justifie leur utilisation pour la résolution de problèmes difficiles.

Des résultats prometteurs...

Les résultats obtenus doivent encore être validés et confrontés à d'autres problèmes, mais ils sont très prometteurs. En particulier, ils ouvrent des perspectives très intéressantes quand au déploiement de nouveaux outils :

- intégrant de nombreux degrés de libertés ;
- gérant les interactions entre métiers, et composants du produit ;
- favorisant ainsi une nouvelle manière d'aborder la conception de produits complexes en explorant/exploitant/expliquant des compromis disciplinaires.

Publications

– J-B. Welcomme and M-P. Gleizes. Résolution de problèmes multidisciplinaires multi-objectifs par systèmes multi-agents adaptatifs : Application à la conception préliminaire avion. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, Carcassonne, October 2007.

– J-B. Welcomme, M-P. Gleizes and R. Redon. Adaptive Multi-Agent Systems for Multidisciplinary Design Optimisation. In *16th International Conference on Engineering Design (ICED)*, Paris, August 2007, The Design Society.

– J-B. Welcomme, M-P. Gleizes and R. Redon. A Self-Organising Multi-Agent System Managing Complex System Design. In *International Conference on Complex Open Distributed Systems, System and Information Sciences Notes (SIWN)*, Vol. 1, No. 2, July 2007.

– J-B. Welcomme, M-P. Gleizes, R. Redon, and T. Druot. Self-Regulating Multi-Agent System for Multi-Disciplinary Optimisation. In *4th European Workshop on Multi-Agent Systems (EUMAS)*, Lisbon, December 2006.

– J-B. Welcomme, M-P. Gleizes and R. Redon. Agent-Aided Preliminary Aircraft Design. In *9th International Design Conference (DESIGN)*, Dubrovnik, May 2006, The Design Society.

– J-B. Welcomme and R. Redon. Multi-Agent System for the Simulation of an Aircraft Design Process. In *19th European Conference on Modelling and Simulation (ECMS)*, Riga, June 2005.

Bibliographie

[1] I. Alaya, C. Solnon, and K. Ghedira. Ant Colony Optimization for Multi-objective Optimization Problems. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 450–457. IEEE Computer Society, 2007.

[2] N. Alexandrov and R-M. Lewis. Recongurability in MDO Problem Synthesis, Part1. In *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, 2004.

[3] J-T. Allison. Complex System Optimization : A Review of Analytical Target Cascading, Collaborative Optimization, and Other Formulations. Master's thesis, Department of Mechanical Engineering, University of Michigan, 2004.

[4] A. Armstrong and E.H. Durfee. Dynamic prioritization of complex agents in distributed constraint satisfaction problems. In *15th International Joint Conferences on Artificial Intelligence (IJCAI 1995)*, pages 620–625, 1997.

[5] C. Badufle. *Définition Conceptuelle d'Avions : Vers une Optimisation Multiobjectifs, Robuste et Incertaine*. PhD thesis, Université Paul Sabatier, 2007.

[6] C. Badufle, C. Blondel, T. Druot, and M. Duffau. Automatic Satisfaction of Constraints Set in Aircraft Sizing Studies. In *6th World Congresses of Structural and Multidisciplinary Optimization (WCSMO'05)*, 2005.

[7] F. Bellifemine, G. Caire, T. Truco, and G. Rimasa. *JADE Administrators's Guide*, 2002.

[8] C. Bernon, V. Camps, M-P. Gleizes, and G. Picard. Designing Agents' Behaviours within the Framework of ADELFE Methodology . In *4th International Workshop on Engineering Societies in the Agents World (ESAW 2003)*, pages 311–327. Springer-Verlag, octobre 2003.

[9] C. Bernon, M-P. Gleizes, S. Peyruqueou, and G. Picard. ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering . In *International Workshop on Engineering Societies in the Agents World (ESAW-2002)*, pages 156–169. Springer-Verlag, septembre 2003.

[10] A. Berro. *Optimisation multiobjectif et stratégie d'évolution en environnement dynamique*. PhD thesis, Université Paul Sabatier, 2001.

[11] E. Bowring, M. Tambe, and M. Yokoo. Multiply-Constrained Distributed Constraint Optimization. In *International Conference on Autonomous Agents and Multiagent Systems AAMAS*, pages 1413–1420. ACM Press, 2006.

[12] B. Brochtrup and J. Herrmann. A Classification Framework For Product Design Optimization. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference IDETC/CIE*, Philadelphia, 2006.

[13] L-T. Bui, J. Branke, and H. Abbass. Multiobjective Optimization for Dynamic Environments. In *Congress on Evolutionary Computation*, Piscataway, NJ, 2005. IEEE Press.

[14] L. Cagnina, S. Esquivel, and C-A. Coello Coello. A Particle Swarm Optimizer for Multi-Objective Optimization. *Journal of Computer Science and Technology JCST*, 5(4), 2005.

[15] V. Camps, M-P. Gleizes, and P. Glize. Une théorie des phénomènes globaux fondée sur des interactions locales . In *6ième Journées Francophones des Systèmes Multi-Agents JFSMA*, pages 207–220. Hermès, janvier 1998.

[16] D. Capera, M.-P. Gleizes, and P. Glize. Mechanism Type Synthesis based on Self-Assembling Agents. 18(9-10) :921–936, 2004.

[17] Capera, D. and Georgé, J-P. and Gleizes, M-P. and Glize, P. The AMAS Theory for Complex Problem Solving Based on Self-Organizing Cooperative Agents. In *First International Workshop on Theory and Practice of Open Computational Systems TAPOCS*, pages 383–388. IEEE, 2003.

[18] C. Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms : A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12) :1245–1287, 2002.

[19] Y. Collette and P. Siarry. *Optimisation multiobjectif*. Eyrolles, 2002.

[20] D. Corne, J-D. Knowles, and M-J. Oates. The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation. In *6th International Conference on Parallel Problem Solving from Nature*, pages 839–848, London, UK, 2000. Springer-Verlag.

[21] Pheonix Corporation. Design Exploration and Optimization Solutions. Technical report, 2004.

[22] K. Deb. Multi-objective Genetic Algorithms : Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3) :205–230, 1999.

[23] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimisation : NSGA-II. In *6th International Conference on Parallel Problem Solving from Nature*, pages 849–858, London, UK, 2000. Springer-Verlag.

[24] J-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The Dynamics of Collective Sorting Robot-Like Ants and Ant-Like Robots. In

First International Conference on Simulation of Adaptive Behavior on From Animals to Animats, pages 356–363, Cambridge, MA, USA, 1990. MIT Press.

[25] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-Organization in Multi-Agent Systems. *The Knowledge Engineering Review*, 20(2) :165–189, 2005.

[26] G. Donnadieu, D. Durand, D-N. Emmanuel Nunez, and L. Saint-Paul. L'approche systémique : de quoi s'agit-il ? *Diffusion de la pensée systémique*, 2003. Synthèse des travaux du Groupe AFSCET.

[27] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2003.

[28] Du, X. and Chen, W. An Efficient Approach to Probabilistic Uncertainty Analysis in SimulationBased Multidisciplinary Design. In *38th AIAA Aerospace Sciences Meeting and Exhibit*, 2000.

[29] D. Durand. *La systémique*. Que sais-je ? Puf, 10 edition, 2002.

[30] E-H. Durfee, V-R. Lesser, and D-D. Corkill. Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transaction on Computers*, 36(11) :1275–1291, 1987.

[31] B. Faltings and M. Yokoo. Introduction : Special Issue on Distributed Constraint Satisfaction. *Artificial Intelligence*, 161(1-2) :1–5, 2005.

[32] J. Ferber. *Les systèmes multi-agents : vers une intelligence collective*. Informatique, Intelligence Artificielle. InterÉditions, 1995.

[33] J-P. Georgé. L'émergence. Rapport de recherche 2003-12-R, IRIT, Université Paul Sabatier, Toulouse, juillet 2004.

[34] J-P. Georgé, M-P. Gleizes, and P. Glize. Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie des AMAS. *Intelligence Artificielle*, 17(4) :591–626, 2003.

[35] M-P. Gleizes and P. Glize. ABROSE : Multi Agent Systems for Adaptive Brokerage. In *4th International Bi-Conference Workshop on Agent- Oriented Information Systems AOIS*, Toronto, Canada, mai 2002.

[36] M-P. Gleizes, P. Glize, and J. Link-Pezet. An Adaptive Multi-Agent Tool For Electronic Commerce . In *Workshop on Knowledge Media Networking, IEEE Ninth International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 2000)*, pages 59–66, Gaithersburg, USA, juin 2000. IEEE Computer Society.

[37] P. Glize. *L'Adaptation des Systemes a Fonctionnalite Emergente par Auto-Organisation Cooperative*. Habilitation à diriger des recherches, Université Paul Sabatier, Toulouse, France, juin 2001.

[38] P. Glize, M-P. Gleizes, and V. Camps. Coopérer pour apprendre sans présupposer. In *Conference on 'Learning : from natural principles to artificial methods*, Genève, juin 1997.

[39] I. Hatzakis and D. Wallace. Dynamic Multi-Objective Optimization with Evolutionary Algorithms : a Forward-Looking Approach. In *8th annual conference on Genetic and Evolutionary Computation GECCO*, volume 2, pages 1201–1208, Seattle, Washington, USA, 2006. ACM Press.

[40] F. Heylighen, P. Cilliers, and C. Gershenson. Complexity and Philosophy, 2007.

[41] F. Heylighen and C. Joslyn. Cybernetics and Second Order Cybernetics. *Encyclopedia of Physical Science and Technology*, 4 :155–170, 2001.

[42] K. Hirayama and M. Yokoo. Distributed Partial Constraint Satisfaction Problem. In *Principles and Practice of Constraint Programming*, pages 222–236, 1997.

[43] K. Hirayama and M. Yokoo. Local Search for Distributed SAT with Complex Local Problems. In *First International Conference on Autonomous Agents and Multi Agent Systems AAMAS*, pages 1199–1206, Bologna, Italy, 2002. ACM Press.

[44] K. Hirayama and M. Yokoo. The distributed breakout algorithms. *Artificial Intelligence*, 161(1-2) :89–115, 2005.

[45] P. Horn. Autonomic computing : Ibm's perspective on the state of information technology. 2001.

[46] K F. Hulme. *The Design of a Simulation-based Framework for the Development of Solution Approaches in Multidisciplinary Design Optimization*. PhD thesis, University of New York at Buffalo, 2000.

[47] M-D. Johnson and K. Rokhsaz. Using Artificial Neural Networks and Self-Organizing Maps for Detection of Airframe Icing. *Journal of Aircraft*, 38(2) :224–230, 2001.

[48] S. Kauffman. *At Home in the Universe : The Search for Laws of Self-Organization and Complexity*. Oxford University Press, October 1996.

[49] H-M. Kim, D-G. Rideout, P-Y. Papalambros, and J-L. Stein. Analytical Target Cascading in Automotive Vehicle Design. *Journal of Mechanical Design*, 125(3) :481–489, 2003.

[50] K. Klamroth and K. Miettinen. Integrating Approximation and Interactive Decision Making in Multicriteria Optimization. *Operations Research*, 2007. à paraitre.

[51] P-N. Koch, J-P. Evans, and D. Powell. Optimization by Simulated Annealing. *Journal Structural and Multidisciplinary Optimization*, 23(2) :111–126, 2002.

[52] J-L LeMoigne. *La modélisation des systèmes complexes*. Dunod, 1990.

[53] R. Mailler and V. Lesser. Solving Distributed Constraint Optimization Problems Using Cooperative Mediation. In *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2004)*, pages 438–445. ACM, 2004.

[54] J-P. Mano. *Etude de l'émergence fonctionnelle au sein d'un réseau de neur-agents coopératifs*. PhD thesis, Université Paul Sabatier, Toulouse, France, 2006.

[55] H. R. Maturana and F. J. Varela. *Autopoiesis and Cognition : The Realization of the Living*. Springer, August 1991.

[56] J-P. Meinadier. *Ingénierie et intégration des systèmes*. Etudes & logiciels informatiques. Broché, 1998.

[57] P. Millot. Supervision et coopération homme-machine : Approche système. *Ingénierie Cognitive IHM et cognition*, pages 191–221, 2003.

[58] P-J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT : Asynchronous Distributed Constraint Optimization with Quality Guarantees. *Artificial Intelligence*, 161(2) :149–180, 2005.

[59] E. Morin. *La méthode : La vie de la vie*. Le seuil, 1980.

[60] E. Morin and J-L. LeMoigne. *L'intelligence de la complexité.* Harmattan, 1999.

[61] K. Ottens, M-P. Gleizes, and P. Glize. A Multi-Agent System for Building Dynamic Ontologies. In *International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS*, pages 1278–1284, Hawaii, USA, mai 2007. ACM Press.

[62] V. Pareto. *Cours d'économie politique*, volume 1-2. F. Rouge, 1896.

[63] H. V. D. Parunak, S-A. Brueckner, R. Matthews, and J. Sauter. How to Calm Hyperactive Agents. In *International Conference on Autonomous Agents and Multi-Agent Systems AAMAS*, pages 1092–1093, New York, NY, USA, 2003. ACM Press.

[64] H. V. D. Parunak, A. Ward, M. Fleischer, J. Sauter, and T-C. Chang. Distributed Component-Centered Design as Agent-Based Distributed Constraint Optimization. In *AAAI Workshop on Constraints and Agents*, pages 93–99. ACM Press, 1997.

[65] V. Pediroda and C. Poloni. Robust Design, Approximation Methods and Self Organizing Map, Techniques for MDO Problems. In *VKI lecture series : Introduction to Optimization and Multidisciplinary Desig*, 2006.

[66] G. Picard. *Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente*. PhD thesis, Université Paul Sabatier, Toulouse, France, décembre 2004.

[67] C. Régis, T. Sontheimer, M-P. Gleizes, and P. Glize. STAFF : un système multi-agent adaptatif en prévision de crues. In *10ièmes Journées Francophones d'Intelligence*

Artificielle Distribuée et des Systèmes Multi-Agents, pages 87–98. Hermès, octobre 2002.

[68] U. Richter, M. Mnif, Branke J., C. Mueller-Schloer, and H. Schmeck. Towards a generic observer/controller architecture for Organic Computing. In C. Hochberger and R. Liskowsky, editors, *INFORMATIK 2006 – Informatik fuer Menschen*, volume P-93 of *Lecture Notes in Informatics (LNI)*, pages 112–119. Bonner Koellen, 2006.

[69] A. Rummler and T. Strufe. Evolvica - A Framework for Evolutionary Computation.

[70] J-D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *First International Conference on Genetic Algorithms*, pages 99–100, 1985.

[71] M-C. Silaghi and M. Yokoo. Nogood Based Asynchronous Distributed Optimization (ADOPT ng). In *International Conference on Autonomous Agents and Multi-Agent Systems AAMAS*, pages 1389–1396, Hakodate, Japan, 2006. ACM Press.

[72] H. A. Simon. *The Sciences of the Artificial*. Massachsetts Institute of Technology, 1969.

[73] R. Sterritt. State of the Art : Autonomic Computing. *Innovations Systems and Software Engineering*, 2005.

[74] R-V. Tappeta, J-E. Renaud, and J-F. Rodriguez. An Interactive Multiobjective Optimization Design Strategy for Decision Based Multidisciplinary Design. *Engineering Optimization*, 34(5) :523–544, 2002.

[75] J-B. Welcomme, M-P. Gleizes, and R. Redon. Agent-Aided Preliminary Aircraft Design. In *9th International Design Conference DESIGN*, Dubrovnik, May 2006. The Design Society.

[76] M. Yokoo. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In *First International Conference on Principles and Practice of Constraint Programming (CP 1995)*, pages 88–102. Springer-Verlag, 1995.

[77] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem : Formalization and algorithms. *IEEE Transaction Knowledge Data Engineering*, 10(5) :673–685, 1998.

[78] J-L. Zhou, A-L. Tits, and Lawrence C-T. *User's Guide for FFSQP Version 3.7 : A FORTRAN Code for Solving Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality and Linear Constraints*. Institute for Systems Research, University of Maryland, 1997.

Résumé

La conception préliminaire avion est un problème d'optimisation multi-disciplinaire et multi-objectif qui consiste à trouver les valeurs des paramètres de description d'un avion et de ses performances en fonction de contraintes. Ces valeurs sont calculées grâce à des connaissances disciplinaires. Les interdépendances entre les paramètres et les non linéarités complexifient les calculs pour lesquels les méthodes classiques se montrent insuffisantes. Pour pallier ce problème, nous proposons une solution basée sur les systèmes multi-agents adaptatifs dans lesquels les agents représentent les disciplines, les paramètres de conception et les performances de l'avion. Les agents en utilisant des connaissances disciplinaires et par un comportement coopératif trouvent collectivement les valeurs des paramètres de conception qui satisfont les contraintes et les performances. Nous montrons que cette approche permet aussi d'intégrer plusieurs questions telles l'étude de sensibilité des paramètres, les fronts de Pareto ou la co-conception en temps réel.

Mots-clés: Adaptation, systèmes multi-agents, optimisation multi-disciplinaire, applications

Abstract

The preliminary design aircraft can be viewed as a multi-disciplinary, multi-objective optimisation problem which consists in finding the values of the aircraft description parameters and of its performances. These values are calculated with disciplinary knowledge. The interdependencies between the parameters and non linearities lead to complex calculus which cannot be realized efficiently by classicmethods. We propose an approach based on adaptive multiagent systems in which agents represent the disciplines, the design parameters and the aircraft performances. In using disciplinary knowledge and a cooperative behaviour, the agents collectively reach the parameters values satisfying constraints and performances. We show this approach affords interesting capabilities such as sensibilities analysis, Pareto front or dynamic interactions.

Keywords: Adaptation, Behaviour Model